Razvoj web aplikacije za iznajmljivanje stanova

Buršić, Petra

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike u Puli

Petra Buršić

Razvoj web aplikacije za iznajmljivanje stanova

Diplomski rad

Pula, rujan 2019. godine

Sveučilište Jurja Dobrile u Puli

Fakultet Informatike u Puli

Petra Buršić

Razvoj web aplikacije za iznajmljivanje stanova

Diplomski rad

JMBAG: 0303054152, redovni student

Studijski smjer: Informatika

Predmet: Napredni algoritmi i strukture podataka

Znanstveno područje: Društvene znanosti

Znanstveno polje: Informacijske i komunikacijske znanosti

Znanstvena grana: Informacijski sustavi i informatologija

Mentor: doc. dr. sc. Tihomir Orehovački

Pula, rujan 2019. godine

IZJAVA O AKADEMSKOJ ČESTITOSTI

Ja, dolje potpisana Petra Buršić, kandidat za magistra informatike ovime izjavljujem da je ovaj Diplomski rad rezultat isključivo mojega vlastitog rada, da se temelji na mojim istraživanjima te da se oslanja na objavljenu literaturu kao što to pokazuju korištene bilješke i bibliografija. Izjavljujem da niti jedan dio Diplomskog rada nije napisan na nedozvoljen način, odnosno da je prepisan iz kojega necitiranog rada, te da ikoji dio rada krši bilo čija autorska prava. Izjavljujem, također, da nijedan dio rada nije iskorišten za koji drugi rad pri bilo kojoj drugoj visokoškolskoj, znanstvenoj ili radnoj ustanovi.

Student

U Puli, 16. rujna, 2019. godine

IZJAVA

o korištenju autorskog djela

Ja, Petra Buršić dajem odobrenje Sveučilištu Jurja Dobrile u Puli, kao nositelju prava iskorištavanja, da moj završni rad pod nazivom „Razvoj web aplikacije za iznajmljivanje stanova“ koristi na način da gore navedeno autorsko djelo, kao cjeloviti tekst trajno objavi u javnoj internetskoj bazi Sveučilišne knjižnice Sveučilišta Jurja Dobrile u Puli te kopira u javnu internetsku bazu završnih radova Nacionalne i sveučilišne knjižnice (stavljanje na raspolaganje javnosti), sve u skladu s Zakonom o autorskom pravu i drugim srodnim pravima i dobrom akademskom praksom, a radi promicanja otvorenoga, slobodnoga pristupa znanstvenim informacijama.

Za korištenje autorskog djela na gore navedeni način ne potražujem naknadu.

U Puli, 16. rujna 2019.

Potpis

_____

Sadržaj

1. Uvod

Tema diplomskog rada je razvoj web aplikacije za iznajmljivanje stanova. Radi se o aplikaciji koja omogućuje privatnim iznajmljivačima oglašavanje iznajmljivih nekretnina koje posjeduju. Kroz aplikaciju iznajmljivači dobivaju iskustvo stvaranja dobro strukturiranog oglasa koji sadrži sve relevantne specifikacije nekretnine. Podstanari tako dobivaju detaljan uvid u iznajmivu nekretninu što znatno olakšava proces donošenja odluka.

Web aplikacija naziva se "Flatly" te u svojoj prvoj iteraciji namijenjena je Pulskom tržištu gdje postoji široka ponuda i potražnja iznajmljivih nekretnina na duži period. Tržište u ovom trenutku nema implementirano rješenje za takav problem te se u tu svrhu koriste različiti alati i oglasnici koji ne pružaju sve potrebne funkcionalnosti te nisu na nikakav način regulirani.

U drugom poglavlju razmatra se motivacija implementacije digitalnog proizvoda te mogućnosti koje tržište donosi.

Treće poglavlje opisuje korištene tehnologije u razvoju aplikacije koje uključuju Angular 6 kao front-end framework za razvoj responzivnih web aplikacija, Node.js u Express framework-u kao server-side biblioteku za komunikaciju između klijentske strane i baze podataka PostgreSQL.

Specifikacija je opisana u četvrtom poglavlju što uključuje analizu zahtjeva i modeliranje sustava, te prototipiranje kroz jednostavan i intuitivan dizajn korisničkog sučelja koji predstavlja samo iskustvo korisnika.

U petom poglavlju, oblikovanje i implementacija, opisana je arhitektura razvoja projekta te na koji se način različiti pod-sustavi spajaju kako bi stvorili funkcionalni sustav koji efikasno rukuje s podacima na strukturirani način. Implementacija opisuje razvoj aplikacije koja se temelji na komponentama unutar jedne web stranice, te struktura baze podataka koja s klijentom komunicira kroz REST API. Radi se o pod-sustavima koji se integriraju i omogućuju efektivan rad aplikacije na svim razinama.

Šesto poglavlje predstavlja konačan proizvod, njegov izgled i funkcionalnosti te korisničke upute koje pokazuju na koji će način korisnik koristiti aplikaciju te što će dobiti kao rezultat njegovih operacija.

U zaključku razmatra se korisnost proizvoda te buduća poboljšanja istoga.

Projekt se razvija konceptualno i implementacijski stvarajući tako stvaran prototip aplikacije spreman za daljnji razvoj i evoluciju. Izvorni kod aplikacije dostupan je na GitHub-u:

https://github.com/pbursic/NajamStanova

2. Motivacija

Danas gotovo svatko tko posjeduje nekretninu može izdvojiti stan ili sobu za iznajmljivanje. Ponuda stanova, koji nisu namijenjeni isključivo turizmu, se tako sve više povećava zajedno sa potražnjom. Ljudi vode fleksibilniji način života gdje često mijenjaju radna mjesta i lokacije pa podstanarstvo predstavlja najbolje rješenje.

Konkretno, prateći Pulsko tržište, uz otvaranje novih studijskih smjerova, mogućnost zapošljavanja u turizmu te rast poduzeća na području koji otvaraju nova radna mjesta u raznim sektorima, potražnja za stanovima bit će sve veća te potrebno je takvo tržište regulirati i konsolidirati na jedno web mjesto.

Najkorištenije rješenje predstavljaju razne Facebook grupe, Njuškalo, Butiga ili oglasne ploče na raznim lokacijama. Moguću prijetnju predstavlja rješenje u razvoju „Stapp“, aplikacija za iznajmljivanje i ocjenjivanje stanova namijenjena isključivo studentima. Postojeća rješenja nisu praktična ukoliko sužavaju potražnju na samo jedan dio tržišta, a to su studenti, ili ne predstavljaju funkcionalno rješenje poput raznih Facebook grupa gdje oglasi često nisu strukturirani te ih nije moguće efikasno pretraživati.

Razmatrajući konkurentska rješenja i prijetnje, moguće je odabrati strategiju minimizacije slabosti i prijetnje kako bi se stavio naglasak na same snage projekta. Snage predstavljaju razvoj aplikacije stavljajući na prvo mjesto iskustvo korisnika, velika potražnja i skalabilnost kao što je prikazano u SWOT analizi u nastavku.

Slika 2.1 SWOT analiza

Aplikacija „Flatly“ služi za oglašavanje i pronalaženje iznajmljivih stanova na određenom području koji zadovoljavaju postavljene kriterije. Radi se o oglasniku gdje korisnici mogu pristupiti u ulozi iznajmljivača i podstanara koji traže smještaj na duže vrijeme. Bitno svojstvo aplikacije je da korisnici mogu čuvati svoje oglase na računu te upravljati njima postavljajući ih na tržište ili povlačeći iste, što znači da korisnik prilikom oglašavanja svojeg objekta neće morati ponovno pisati novi oglas već može iskoristiti prethodno stvoreni. Svaki oglas se sastoji od mnogih specifikacija koje detaljno opisuju iznajmljivu nekretninu čime iznamljivaći dobivaju uvid u sve bitne informacije koje bi trebali pružiti potražnji kako bi olakšali pretraživanje te efektivno ostvarili svoj cilj.

Glavne prednosti za iznajmljivače predstavlja prilagođeno sučelje za stvaranje oglasa te pohrana oglasa na vlastitom računu koji mogu imati status aktivan ili neaktivan što znači da prilikom oglašavanja korisnik ne mora ponovno stvarati novi oglas već može iskoristiti postojeći sa željenim promjenama. Za podstanare glavnu prednost predstavlja sučelje koje omogućuje jednostavno i detaljno pretraživanje stanova pružajući sve bitne informacije za pronalaženje savršenog prostora.

3. Korištene tehnologije

3.1. Angular

Angular je JavaScript okvir (eng. framework) za razvoj prednjeg ureda (eng. front-end development) koji se koristi za razvoj dinamičkih web aplikacija za desktop i mobilne uređaje (Angular 2019).

Radi se o okviru otvorenog koda kojega razvijaju Google inženjeri. Google je službeno objavio prvu verziju, AngularJS, 2012. godine i od tada ga održava (Hostinger 2019).

Web razvoj prednjeg ureda jest programiranje grafičkog sučelja kako bi korisnik mogao pregledavati i komunicirati s podacima unutar aplikacije. Korišteni jezici za razvoj su HTML, CSS i JavaScript.

SPA je skraćenica za aplikacije na jednoj stranici (eng. Single Page Applications). Njihova glavna karakteristika jest prikazivanje sadržaja na jednoj glavnoj stranici što znači da prilikom navigacije korisnika kroz aplikaciju nema potrebe za slanje zahtjeva poslužitelju za dohvačanje novog HTML-a. SPA aplikacije se oslanjaju na usmjerivač (eng. router). Usmjerivači se sastoje od ruta koje opisuju mjesto na kojem bi trebali odgovarati. To mogu biti statični (/posts) ili dinamički (/post-detail/:id, gdje vrijednost :id može biti bilo koji broj mogućnosti, staza) (Pshrmn 2019).

Osnovne funkcije omogućava API (eng. Application Programming Interface) koji omogućava komunikaciju više djelova aplikacije, konkretno prednji i stražnji ured.

Struktura Angular-a temelji se na arhitekturi komponenti i usluga. AngularJS se temeljio na MVC arhitekturi. Konkretno u projektu korištena je verzija Angular 6. Angular 6 je objavljen u svibnju 2018. godine (Tutorialspoint 2019).

U razvoju je korišten materijalni dizajn (eng. Material design), radi se o vizualnom jeziku koji sintetizira klasična načela dizajna s inovacijom tehnologije. Cilj je razviti temeljni sustav koji objedinjuje korisničko iskustvo na svim platformama, uređajima i metodama unosa. Materijalni dizajn inspiriran je fizičkim svijetom i njegovim teksturama, uključujući i način na koji reflektiraju svjetlost i bacaju sjene (Material design 2019).

Angular materijal predstavlja materijalni dizajn za komponente u Angular-u. Omogućuje brz razvoj i dizajn korisničkog sučelja uz moderne komponente koje funkcioniraju na web-u kao i na mobilnim uređajima te stolnim računalima (Angular material 2019).

3.2. Node.js

Node je dizajniran za izgradnju prilagodljivih mrežnih aplikacija (Node.js 2019).

Radi se o biblioteci na strani poslužitelja (eng. server-side) koja omogućuje komunikaciju između klijentske strane i baze podataka.

Node.js je JavaScript runtime okruženje otvorenog koda koji se koristi na raznim platformama. Implementirana aplikacija unutar Node-a se izvodi u jedinstvenom procesu bez kreiranja nove niti (eng. thread) za svaki novi zahtjev (eng. request). Moguće je obraditi tisuće istodobnih veza (eng. connections) s jednim poslužiteljem. Postao je popularan među programerima prednjeg ureda jer nije potrebno učiti novi programski jezik za upravljanjem strane poslužitelja (Node.js Dev 2019).

Konkretno, kroz Node je moguće stvoriti vezu sa bazom, slušati zahtjeve u obliku SQL upita, dohvatiti podatke te poslati odgovore (eng. response) na klijentsku stranu.

3.3. PostgreSQL

PostgreSQL je sustav za upravljanje relacijskom bazom podataka (eng. RDBMS - Relational database management system) otvorenog koda.

PostgreSQL je moćan objektno-relacijski sustav baza podataka koji koristi i proširuje SQL jezik u kombinaciji s mnogim značajkama koje sigurno pohranjuju i skaliraju najkompliciranija podatkovna opterećenja. Počeci PostgreSQL-a datiraju iz 1986. godine kao dio projekta POSTGRES na Sveučilištu California u Berkeleyu i ima više od 30 godina aktivnog razvoja na temeljnoj platformi.

PostgreSQL je stekao snažnu reputaciju zahvaljujući dokazanoj arhitekturi, pouzdanosti, integritetu podataka, robusnom skupu značajki, proširivosti i predanosti open source zajednici iza softvera za dosljedno pružanje učinkovitih i inovativnih rješenja (PostgreSQL 2019).

4. Specifikacija

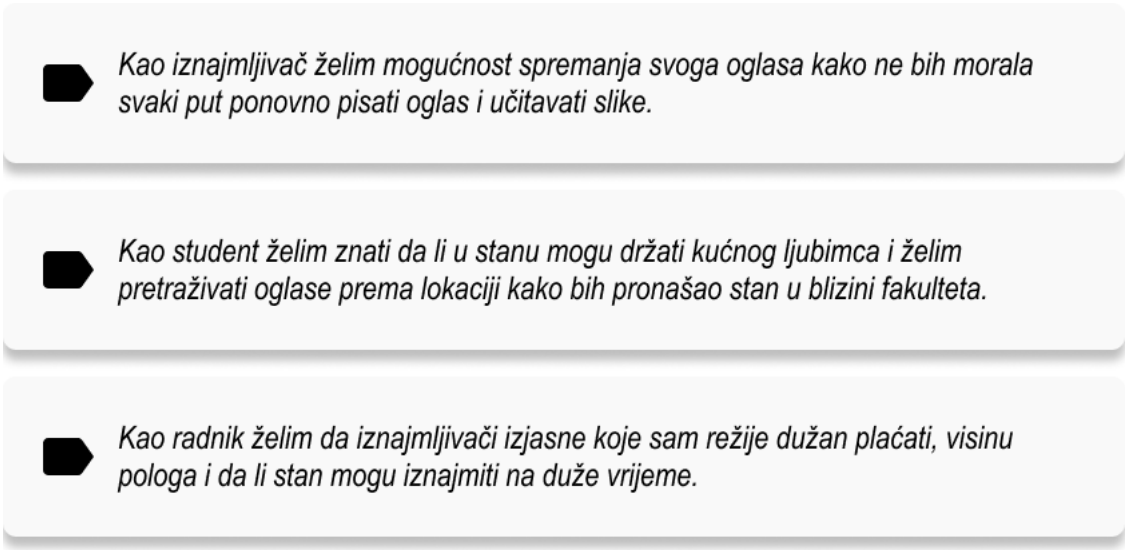
4.1. Modeliranje sustava

Specifikacija je prva osnovna aktivnost modeliranja programskog procesa koja utvrđuje što softver treba raditi. Osim samih razvijača, u proces su uključeni i budući korisnici proizvoda koji opisuju što bi softver trebao raditi i kako će se koristiti (Manger 2005).

Razvijena aplikacija obuhvaća veliki broj i raznoliku vrstu korisnika te nemoguće je sve tipologije korisnika svrstati u jednu kutiju kako bi se definirali zahtjevi. Zahtjevi mogu biti vrlo specifični s obzirom na tipologiju korisnika te potrebno je sakupiti zahtjeve i utvrditi postoji li zajednički obrazac (eng. pattern) među korisnicima.

U tu je svrhu moguće kreirati razne korisničke profile koji će utvrditi zahtjeve ciljanih korsnika. Radi se o korisničkim pričama, a to su kratki i jednostavni opisi svojstva proizvoda sa korisničke perspektive. Korisnici zapravo opisuju sebe i što žele postići korištenjem sustava, jasno se definira cilj i razlog. Korisničke priče često se pišu na indeksnim karticama koje se rasporede na zidu ili stolu kako bi se olakšalo planiranje i poticala rasprava (Mountain Goat Software 2019).

Slika 4.1 dolje u nastavku definira tri korisničke priče koje čine temelje za raspravu i utvrđivanje korisničkih zahtjeva.



Slika 4.1 Korisničke priče

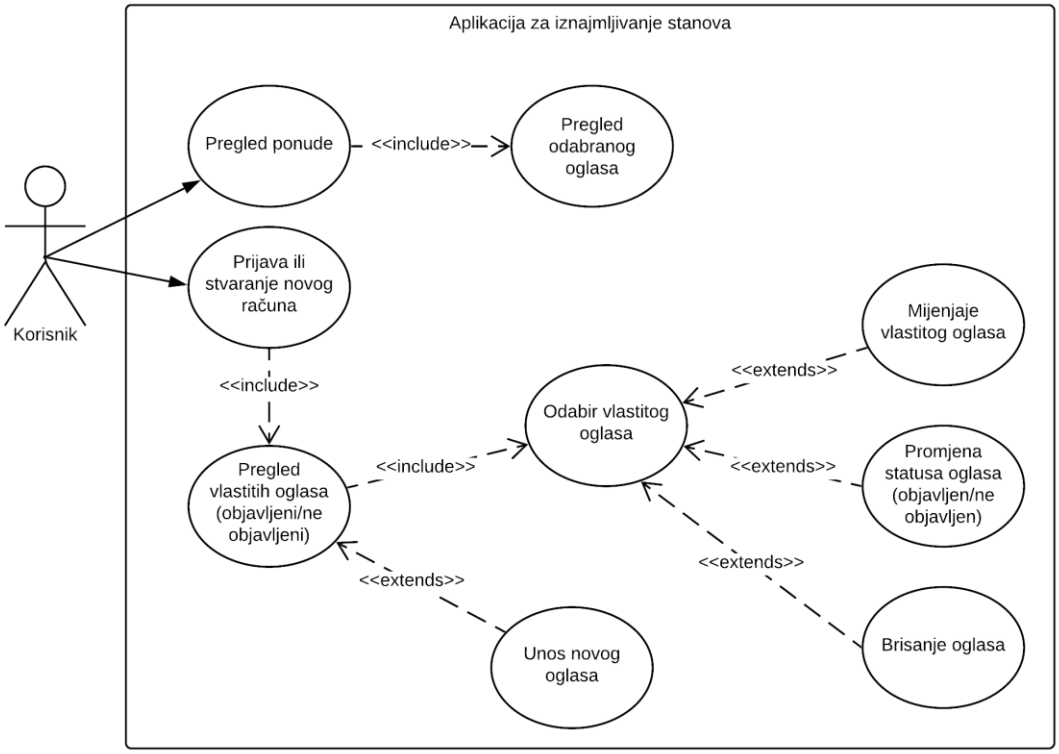
Tijekom analize zahtjeva sastavljaju se modeli budućeg sustava uzimajući u obzir korisničke zahtjeve. Modeli služe za opisati sustav na grafički način koji je razumljiv razvijačima kao i korisnicima. To je bitan korak koji konsolidira viziju razvijača i viziju korisnika kako bi se pronašao kompromis i konsenzus svih strana prije oblikovanja samog sustava.

Modeli su lako razumljivi zato što su pojednostavljeni i daju apstraktnu sliku sustava. Prikazuju ponašanje sustava, reakcije na događaje i promjene stanja, određuje kontekst i granice sustava i njegove okoline, te modelira arhitekturu i građu podataka.

Kako bi se dobila cjelovita slika o sustavu, potrebno ga je promatrati iz različitih perspektiva kroz komplementarne modele (Manger 2005).

U tu je svrhu postavljena standardna notacija i način crtanja dijagrama pod nazivom UML (Unified Modeling Language). Kako bi se prikazali različiti načini na koje korisnik može pristupiti u interakciji sa sustavom koristi se dijagram slučaja uporabe (eng. Use case diagram). Isti sažima pojedinosti korisnika sustava u ulozi aktera i njegove interakcije sa sustavom te interakcije sustava s okolinom. Definira područje i obujam sustava te pomaže akterima da postignu svoje ciljeve korištenjem aplikacije (Lucidchart 2019).

Način na koji korsnik može pristupiti u interakciju sa sustavom prikazano je dijagramom slučaja uporabe na slici u nastavku.



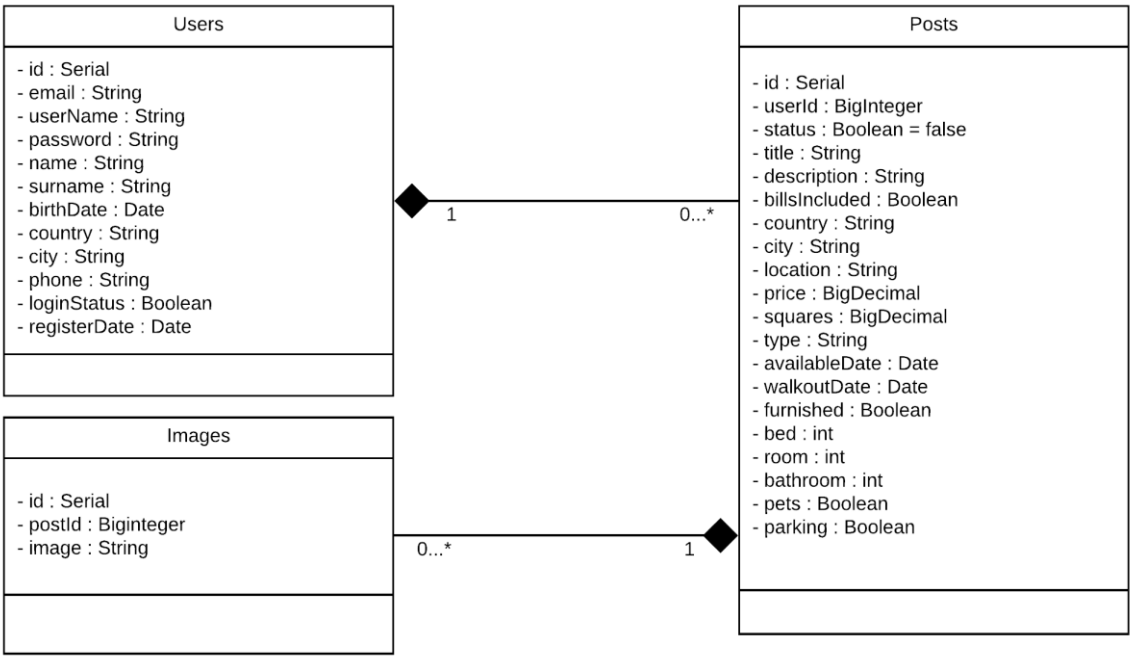
Slika 4.2 Dijagram slučaja uporabe

Korisnik može pristupiti aplikaciji bez potrebe prijave ili registacije računa, međutim, ukoliko korisnik nije prijavljen u sustav, imati će određena ograničenja u korištenju same aplikacije. Naime, korisnik može pregledavati i filtrirati oglase koje ga zanimaju, no neće biti u mogućnosti stupiti u kontakt s najmodavcem ukoliko ne posjeduje vlastiti račun. Podaci kontakt osobe takvom korisniku nisu dostupni.

Nakon prijave ili registracije računa, korisnik dobiva uvid u kontakte najmodavca te otvara mu se mogućnost kreiranja vlastitog oglasa. Obrazac za stvaranje novoga oglasa prilagođen je potrebama najmodavca i uključuje sve relevantne informacije vezane uz nekretninu.

Vlasnik računa može stvarati neograničeni broj oglasa. Svaki se oglas sprema na račun te vlasnik može postaviti oglas na tržište kao aktivan oglas ili ga može sačuvati za buduće oglašavanje. Na taj način vlasnik računa može aktivirati ili deaktivirati svoje oglase po želji bez potrebe ponovnog pisanja oglasa ili učitavanja slika. Svaki se oglas može naknadno urediti ili potpuno izbrisati.

Struktura sistema prikazana je klasnim dijagramom na slici u nastavku.



Slika 4.3 Klasni dijagram

Jedan korisnik može imati niti jedan ili više oglasa prema čemu jedan oglas mora biti u vlasništvu samo jednoga korisnika. Svaki oglas može imati niti jednu ili više slika.

4.2. Prototipiranje korisničkog sučelja

Upotreba prototipa je sastavni dio dizajniranja aplikacije jer omogućuje brzo testiranje ideja kroz više iteracija koje su evaluirane sa strane razvijača kao i sa strane samih korisnika. Korisnici dobivaju još jasniju sliku proizvoda na koju mogu davati primjedbe i prijedloge za poboljšanja prije same implementacije.

Prototipiranje je ključan korak bez kojega se ne može krenuti u implementaciju ukoliko često apstraktni prikazi poput dijagrama mogu biti vrlo općeniti i iako definiraju interakciju korisnika i sustava, samim korisnicima može biti nejasno kako će se oni sami snalaziti u okruženju i da li je njihova vizija usklađena sa vizijom inženjera i razvijača. Često korisnici sustava ne uspijevaju jasno izraziti i definirati zahtjeve jer je njihov način razmišljanja drugačiji od inženjerske vizije.

Prototipiranje ima dvije pod-aktivnosti, a to su otkrivanje zahtjeva i validacija. Prototipovi omogućuju eksperimentiranje gdje korsnici najčešće dobivaju nove ideje o tome kako bi sustav trebao raditi. Isto tako moguće je uočiti greške i propuste prototipa koje otkrivaju sve nesporazume između razvijača softvera i korisnika, te demonstrira se izvedivost i korisnost sustava (Manger 2005).

Prototipiranjem korisničkog sučelja (eng. user interface) se najčešće bave dizajneri koji naglasak ne stavljaju samo na grafičku atraktivnost sučelja već na iskoristivost (eng. usability) i cjelokupno iskustvo korisnika (eng. user experience).

Dizajniranje korisničkog iskustva odgovara na pitanje „Zašto?“, odnosno motivacija i pogledi korsnika. Pritanje „Što?“, funkcionalnosti i značajke sustava, te „Kako?“, odnosno pristupačnost i estetika aplikacije (Interaction design foundation 2019).

Dizajn korisničkog sučelja pomaže korisnicima da ostvare svoj cilj na efikasan način. Dizajn mora biti inovativan i intuitivan (Beocaft 2019).

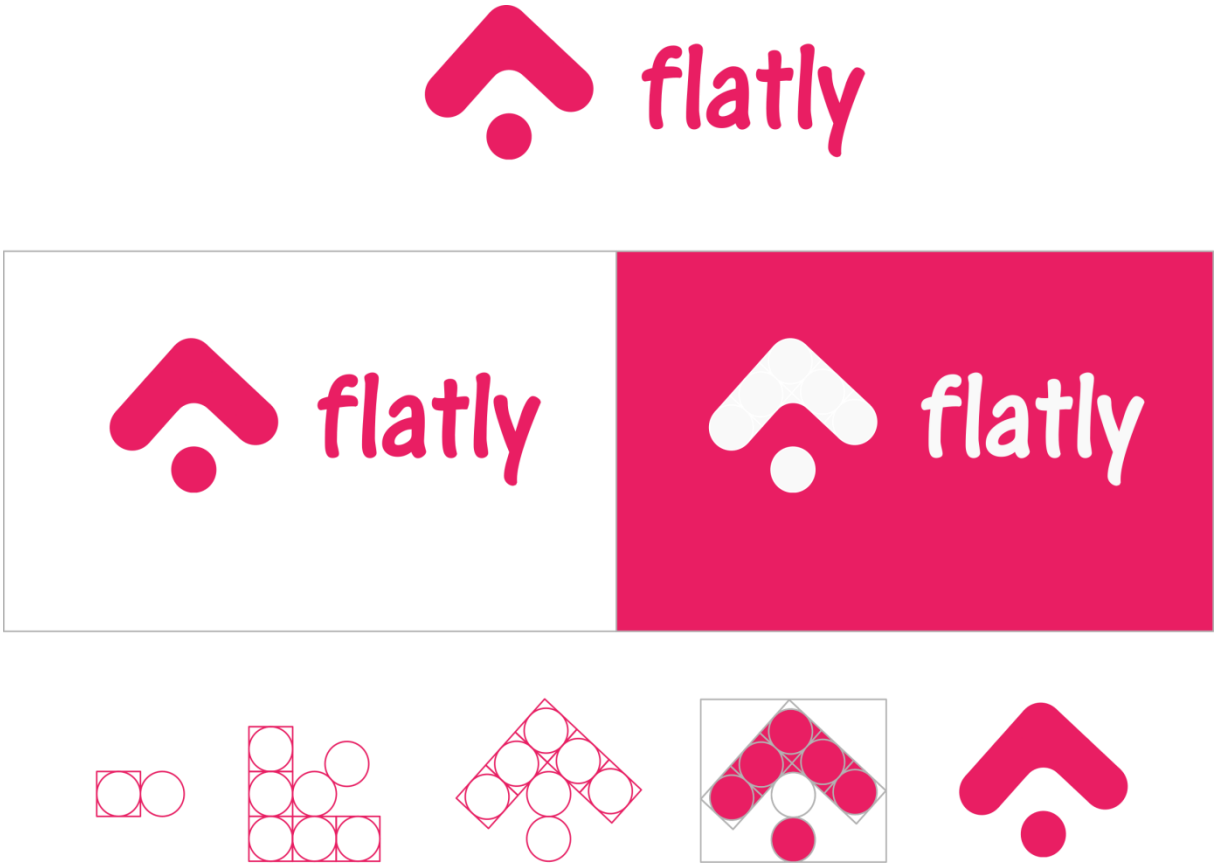
Optimizacija dizajna i prilagođavanje korisničkog sučelja ostvareno je uz pomoć alata kao što su Inkscape i Figma.

Inkscape je profesionalni alat za uređivanje vektorskih grafika. Koriste ga ilustratori i web dizajneri za crtanje i obradu teksta (Inkscape 2019).

U sklopu projekta alat je korišten u svrhu stvaranja vizualnog identiteta i logotipa za web aplikaciju.

Vizualni identitet čine elementi poput:

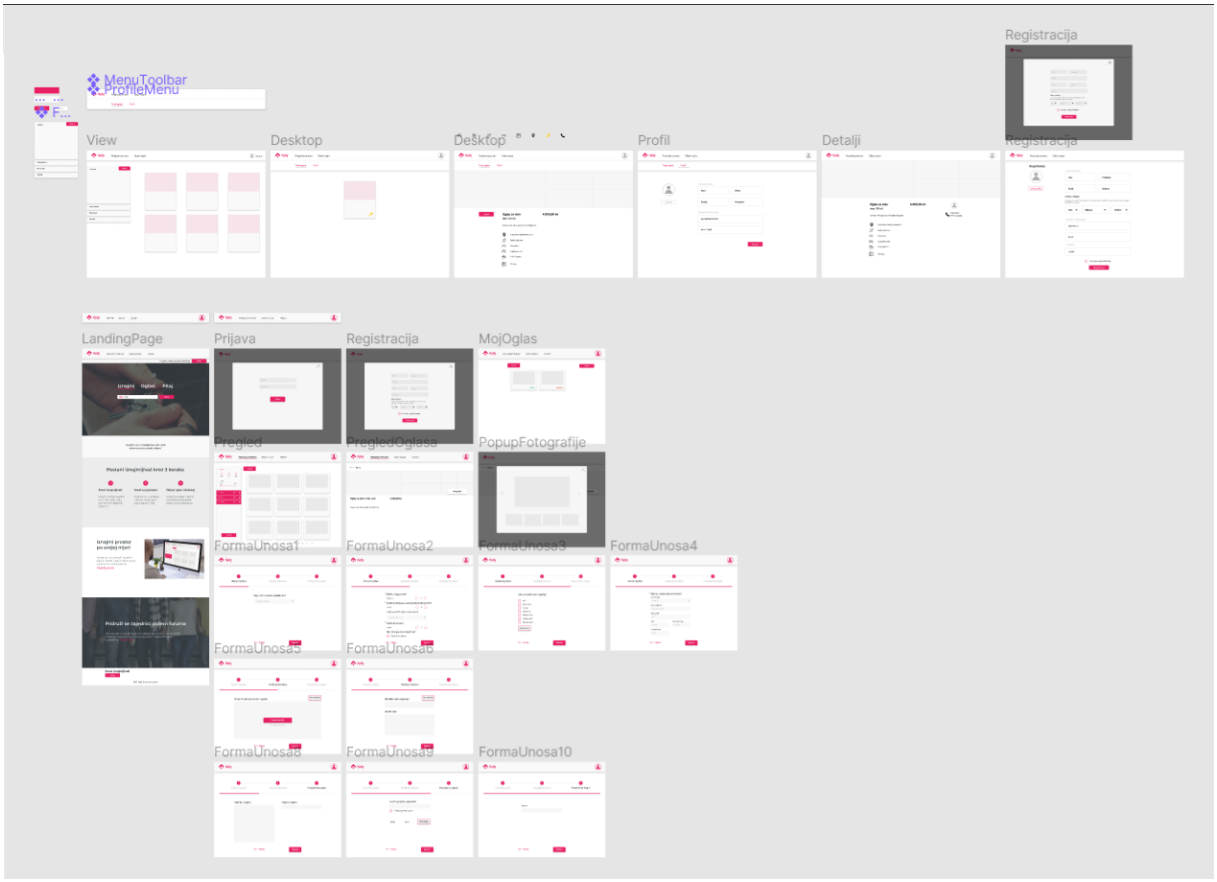
- logotip,
- paleta boja,
- tipografija (Nela Dunato 2019).



Slika 4.4 Vizualni identitet

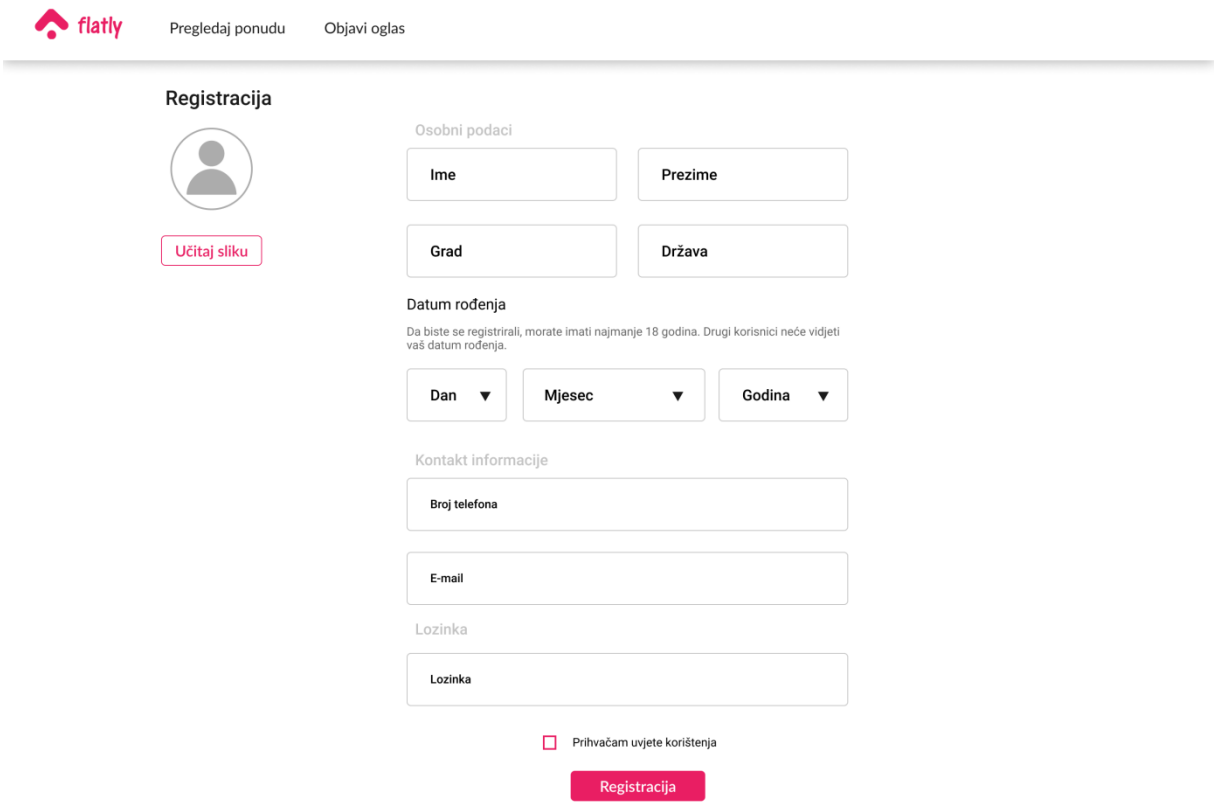
Figma je aplikacija za dizajniranje i prototipiranje korisničkog sučelja (Figma 2019).

Kroz aplikaciju se stvaraju okviri i komponente koje određuju izgled aplikacije, te moguće je ostvariti prototip navigacije korisnika kako bi se simuliralo korisničko iskustvo. Prototip izrađen u Figmi prikazan je na slici u nastavku.

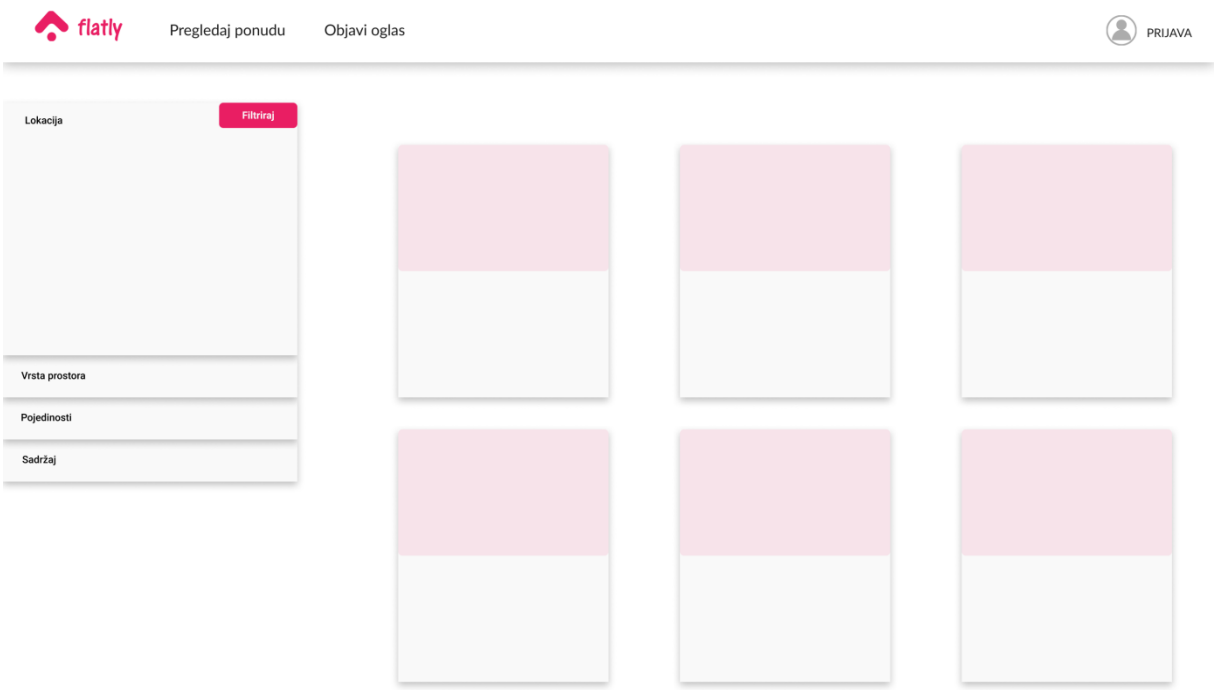


Slika 4.5 Figma prototip

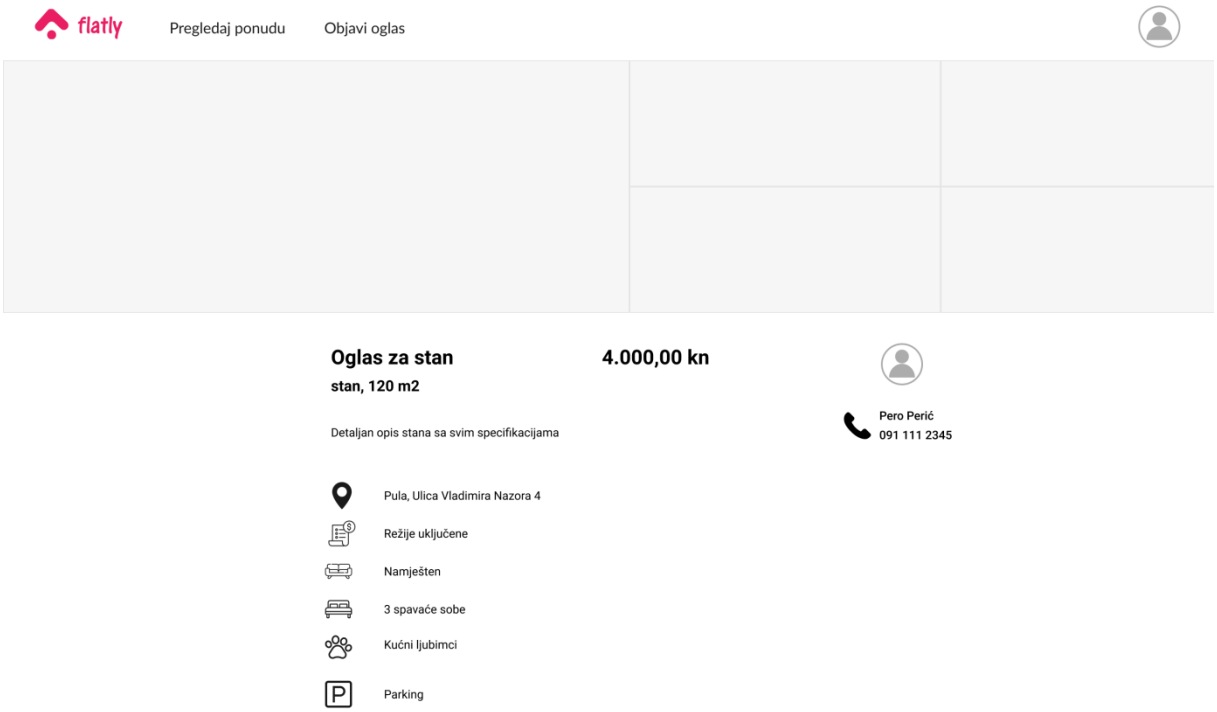
U nastavku su izdvojeni dijelovi prototipa s ključnim ekranima aplikacije.



Slika 4.6 Prototip registracije korisnika



Slika 4.7 Prototip pregleda oglasa



Slika 4.8 Prototip detalja oglasa

5. Oblikovanje i implementacija

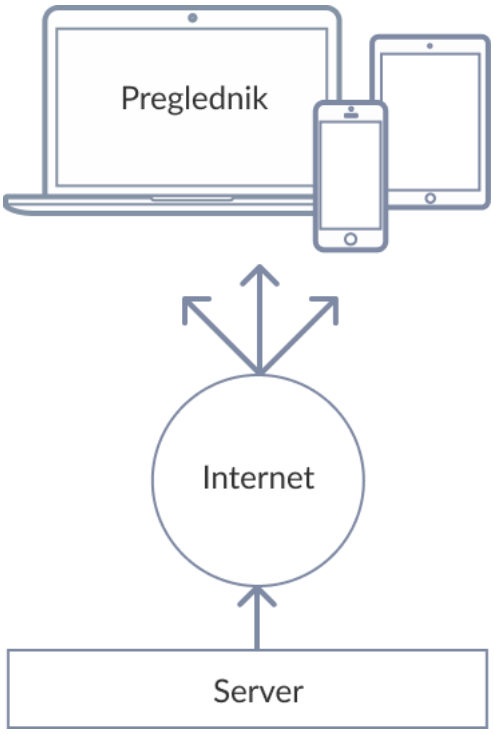
5.1. Arhitektura

Oblikovanje utvrđuje kako će sustav raditi. Radi se o dizajnu ili detaljnom opisu građe sustava. Dizajn sustava je detaljan opis i iterativan proces građe svih dijelova sustava, korisničkog sučelja i strukture podataka (Manger 2005).

Sustav se strukturira sa više pod-sustava koji međusobno komuniciraju i razmjenjuju podatke. Arhitektura sustava klijent-poslužitelj je model koji se satoji od tri dijela:

- poslužitelj,
- klijent i
- mreža.

Na primjeru web aplikacija, poslužitelj je web server, klijent je preglednik poput Google Chrome, a mreža je Internet. Takvu je arhitekturu moguće prikazati dijagramom kao što je vidljivo na slici u nastavku.



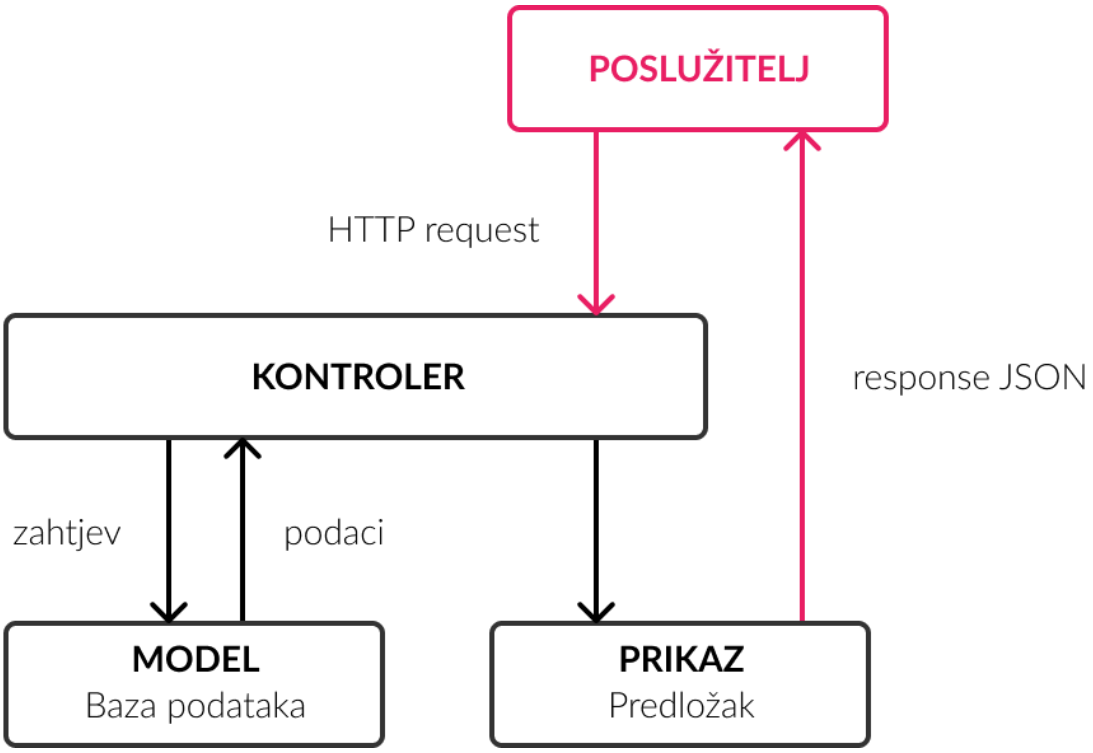
Slika 5.1 Model klijent-poslužitelj

Angular koristi arhitekturu Model-View-Controller (MVC), koja se koristi u razvoju web-aplikacija.

Aplikacija se izgrađuje modularno što znači da se izoliraju funkcionalne jedinice koje se međusobno integriraju kako bi se pojednostavio razvoj i održavanje aplikacije. MVC dizajn je odličan za razvoj web aplikacija jer sam razvoj kombinira nekoliko tehnologija koje se dijele na skup slojeva. Interakcija sa korisnikom je ciklus gdje korisnik poduzima akciju, a kao odgovor, aplikacija mijenja svoj model podataka i vraća korisniku ažurirani pregled. Ovo je vrlo pogodno za web aplikacije isporučene kao niz HTTP zahtjeva i odgovora (Pop i Altar 2014).

Ovaj tip arhitekture sastoji se od tri glavne kategorije:

- model je struktura podataka koja upravlja informacijama i prima ulaz od kontrolera,
- pregled (eng. view) je prikaz informacija na sučelje,
- kontroler (eng. controller) odgovara na ulaz i stupa u interakciju s modelom (Hostinger 2019).



Slika 5.2 MVC arhitektura

Model

Model je dio sustava koji upravlja svim zadacima vezanim uz podatke kao što je provjera valjanosti, stanje sesije i struktura podataka. Sloj modela odgovoran je za poslovnu logiku i obuhvaća metode pristupa podacima i datotekama.

Model se gradi apstrakcijom podataka, validacijom i autentikacijom te se sastoji od klasa koje definiraju domenu interesa. Objekti koji pripadaju domeni često inkapsuliraju podatke koji su pohranjeni u bazama podataka, ali također uključuju i kod koji se koristi za manipuliranje tim podacima (Pop i Altar 2014).

Pregled

Pregledni sloj se sastoji od predložaka i web dizajna. Kontrolira način prikaza podataka i način na koji korisnik vrši interakciju sa sustavom pružajući tako načine za prikupljanje podataka od korisnika. Tehnologije koje se uglavnom koriste su HTML, CSS i JavaScript.

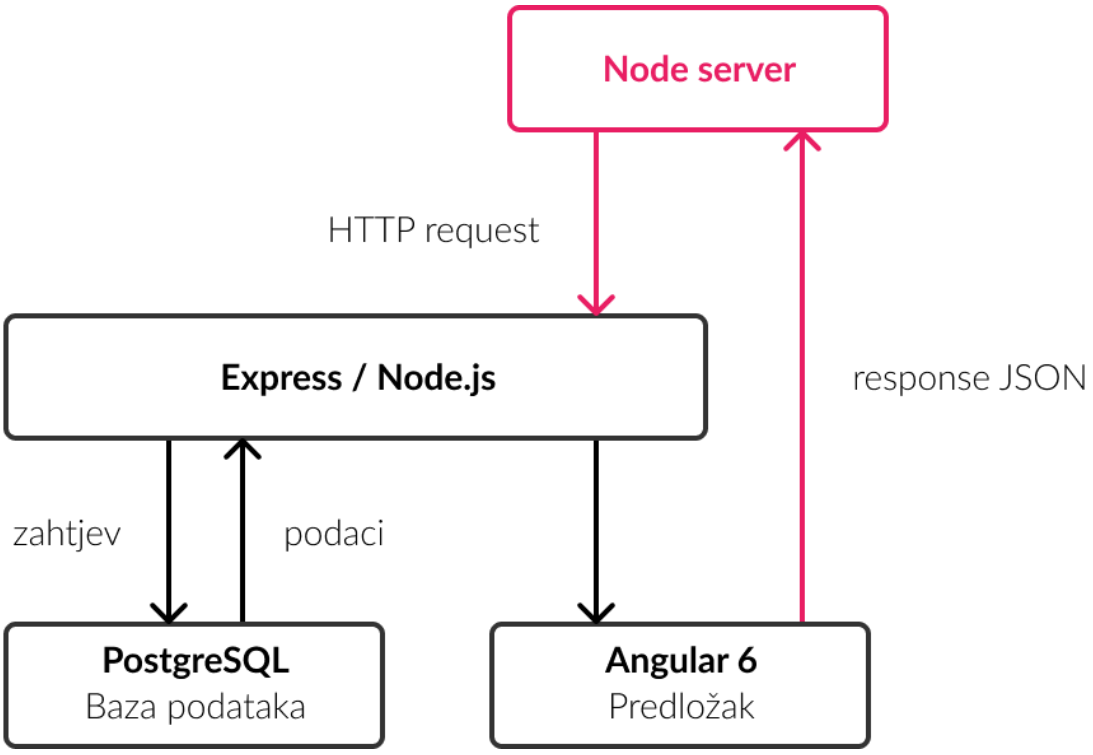
Pregled je odgovoran za upravljanje grafičkim korisničkim sučeljem. To obuhvaća sve forme, gumbovi, grafički elementi i svi ostali HTML elementi koji se nalaze unutar aplikacije (Pop i Altar 2014).

Kontroler

Kontroler je odgovoran za rukovanje događajim (eng. event handling) koje može pokrenuti korisnik ili sustav. Kontroler prihvaća zahtjeve i priprema podatke za slanje odgovora.

Kontroler stupa u interakciju s modelom radi preuzimanja potrebnih podataka i generiranja prikaza u određenom formatu. Zahtjev se zatraži na poslužitelju (eng. server), a MVC okvir šalje metodu u kontroler na temelju URL-a. Isto tako odgovoran je za rukovanje greškama (Pop i Altar 2014).

U kontekstu korištenih tehnologija, arhitektura je sljedeća.

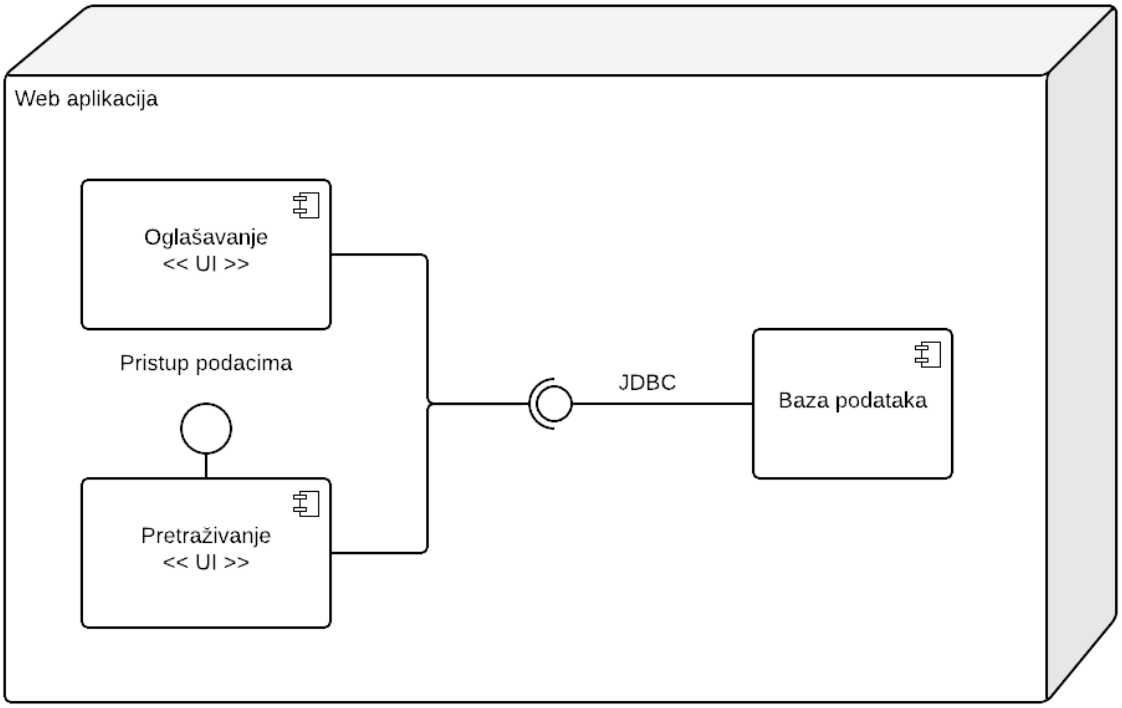


Slika 5.3 MVC tehnologije

5.2. Implementacija

5.2.1. Razvoj temeljen na komponentama

Dijagram komponenti prikazuje odnos između različitih komponenti sustava. Komponenta se odnosi na modul koja predstavlja nezavisne podsustave sa sposobnošću povezivanja s ostatkom sustava (Lucidchart 2019).



Slika 5.4 Dijagram komponenti

Sustav se sastoji od mnogih komponenti, no u dijagramu su prikazane tri ključne komponente sustava od kojih su dvije sučelja. Baza podataka je komponenta koja zahtijeva dva korisnička sučelja za pretraživanje i oglašavanje. Komponenta pretraživanja zahtijeva pristup podacima iz baze podataka kako bi sučelje prikazalo dostupne oglase.

Postoji čitav razvojni pristup koji se vrti oko komponenti, odnosno razvoj temeljen na komponentama (eng. CBS - Component-based development). Radi se o načinu razvoja koji omogućuje modularnost i ponovnu uporabu.

Komponenta je samostalni pružatelj usluga. Kad sustav treba uslugu, on poziva komponentu. Svaka komponenta ima svoje sučelje kroz koje se obavljaju operacije (Manger 2005).

Radi njihove modularnosti, komponente su usluge koje se ponovno upotrebljavaju i jednostavno uključuju unutar sustava. Jedno sučelje može uključivati mnogo komponenti koje pružaju vizualne i funkcionalne usluge za izgradnju pojedinog modula.

5.2.2. Baza podataka

Pohrana podataka ostvarena je kroz PostgreSQL sustav za upravljanje relacijskom bazom podataka. Sastoji se od tri tablice koje su međusobno povezane:

- Tablica "users" - pohranjuje podatke korisnika i njegovog računa
- Tablica "posts" - pohranjuje oglase korisnika
- Tablica "images" - pohranjuje fotografije vezane uz pojedini oglas

Korisnici (eng. users)

Tablica "users" pohranjuje osobne i kontakt podatke korisnika te podatke vezane uz račun korisnika poput elektroničke pošte putem koje se korisnik prijavljuje u aplikaciju te odgovarajuća kriptirana lozinka.

Sastoji se od 11 atributa koji opisuju sva potrebna svojstva za stvoriti sliku korisnika i pružati određenu razinu sigurnosti vezanu uz zaštitu podataka individualca. Atribut "id", tipa "bigint", se jednostavno generira sekvencijano bez ponavljanja brojeva, upravo se zato naziva identifikacijski ključ ukoliko je jedinstven za svaki redak tablice. Sljedeća dva atributa su "email" i "password", elektronička pošta i lozinka koji su potrebni za autentifikaciju korisnika i provjeru postojanja računa. Vezano uz sigurnost lozinka se kriptira heširajućom funkcijom.

U tablicu se spremaju osobni i kontakt podaci vezani uz korisnika. Osobni podaci koji su javno vidljivi ostalim korisnicima koji posjeduju vlastiti račun unutar aplikacije služe kako bi korisnici mogli dospijeti u kontakt međusobno te sklopiti ugovor o najmu. Radi se o atributima "name" - ime, "surname" - prezime, "phone" - broj telefona i "image" - fotografija profila. Dok podaci koji su osobni za korisnika i nisu vidljivi niti dostupni ostalim korisnicima su "birth_date" - datum rođenja, "country" - država stanovanja i "city" - grad stanovanja. Atribut "register-date" pohranjuje datum registracije korisnika i tipa je "timestamp" - vremenska oznaka, pohranjuje se kao datum i vrijeme prve registracije računa. Njegova je zadana vrijednost trenutni datum i vrijeme izvršavanja registacije.

Atribut	Tip podataka	Definicija	Opis
id	bigint	Primary key, Not NULL	identifikacijski ključ
email	character varying		elektronička pošta
password	character varying		lozinka
name	character varying		ime korisnika
surname	character varying		prezime korisnika
birth_date	date		datum rođenja
country	character varying		država stanovanja
city	character varying		grad stanovanja
phone	character varying		telefonski kontakt
registered_date	timestamp without time zone	length: 4, Default value: CURRENT TIMESTAMP	datum registracije korisnik
image	character varying	length: 10,485,760	fotografija korisnika

Tabela 5.1 Tablica users

Oglasi (eng. posts)

Sljedeća je tablica "posts" koja pohranjuje oglase korisnika. Sastoji se od 21 atributa koji detaljno opisuju nekretninu koja se iznajmljuje. Atribut "id" je identifikacijski ključ jedinstven za svaki oglas, nema ponavljanja i sekvencijalno se generira. Dok, atribut "user_id" je vanjski ključ koji spaja tablicu oglasa s tablicom korisnika. Razlog toga je što ne može postojati oglas bez da postoji korisnik koji je vlasnik oglasa. U ovo se polje pohranjuje identifikacijski ključ računa iz tablice korisnika koji je kreirao oglas. Unos oglasa se nikako ne može izvršiti bez odgovarajućeg ključa koji je postojeći u tablici korisnika.

Atribut "status" pokazuje da li je oglas aktivan ili neaktivan. Ovaj podatak je tipa "boolean" koji pohranjuje vrijednosti "true" - istina ili "false" - laž. Podatak se prikazuje samo vlasniku oglasa koji može mijenjati vrijednost polja kroz sučelje aplikacije kako bi postavio vlastiti oglas na tržište ili ga povukao. Na glavnom sučelju aplikacije gdje

je moguće pretraživati oglase, prikazani su samo oni čiji je status "true" jer to označava da je valsnik oglasa u potrazi za podstanarima.

Atribut "address" je polje koje pohranjuje adresu na kojoj se stan nalazi, slično kao i u tablici korisnika, adresa je podatak poput kontakta osobe, odnosno vidljivo je unutar aplikacije samo korisnicima koji posjeduju vlastiti račun.

Ostali atributi su svojstva koja opisuju nekretninu i vidljiva su svim korisnicima koji pristupaju aplikaciji čak iako nemaju vlastiti račun. Atribut "title" i "description" su naslov i opis oglasa koje korisnik unosi kroz sučelje. Definicija polja postavlja ograničenja vezana uz dužinu niza, za naslov dužina je 150 dok je za opis 1,000.

Atributi "country" - država, "city" - grad i "type" - tip nekretnine su nizovi ograničene duljine od 150 znakova. Opisuju državu i grad nekretnine koja se iznajmljuje te njezin tip koji može biti studio apartman, stan, kuća, itd. Nadalje "squares" - površina u m2 je tipa "bigint" dok su "price" - cijena i "deposit" - polog tipa "money" ukoliko se radi o novčanim vrijednostima. Atributi "bills_included" - uključene režije u cijenu, "furnished" - stan namiješten, "pet" - kućni ljubimci dobrodošli, "parking" - dostupno parking mjesto i "wifi" - interent konekcija, su tipa "boolean" gdje unesena vrijednost može biti istina ili laž. Broj soba i kreveta sadržan je u poljima "bed" - krevet, i "room" - soba. Dok, "available_date" - dostupno od datuma, i "walkout_date" dostupno do datuma, upravo ukazuju na dostupnost smještaja.

Atribut	Tip podataka	Definicija	Opis
id	bigint	Primary key, Not NULL	identifikacijski ključ
user_id	bigint	Foreign key, Not NULL	vanjski ključ
status	boolean		aktivnost oglasa
title	character varying	length: 150	naslov oglasa
description	character varying	length: 1000	opis oglasa
bills_included	boolean		uključene režije u cijenu
country	character varying	length: 150	država
city	character varying	length: 150	grad nekretnine
address	character varying	length: 500	adresa nekretnine
price	money		mjesečna najamnina
squares	bigint		površina m2
type	character varying	length: 150	tip nekretnine
available_date	date		dostupno od
walkout_date	date		dostupno do
furnished	boolean		stan namješten
bed	bigint		kreveti
room	bigint		sobe
pet	boolean		kućni ljubimci
parking	boolean		parking mjesto
wifi	boolean		internet mreža
deposit	money		polog

Tabela 5.2 Tablica posts

Tablica "images" pohranjuje fotografije za određeni oglas koji je u vlasništvu korisnika.

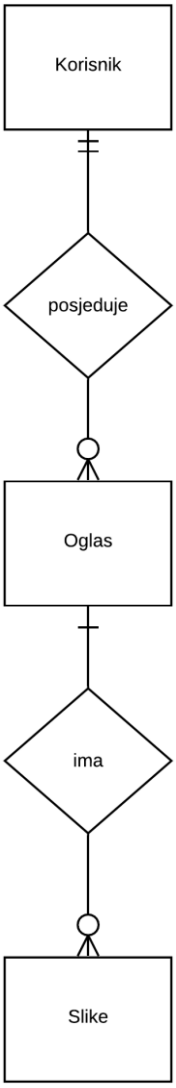
Atribut "id" predstavlja identifikacijski ključ i to je primarni ključ tablice, dok je "post_id" vanjski ključ tablice ukoliko je to ključ koji povezuje tablicu "images" sa tablicom "posts". Redak u tablici fotografija ne može postojati ako ne postoji oglas kojemu te fotografije pripadaju. Slike se spremaju kao niz znakova te se učitavaju na klijentskoj strani.

Atributi	Tip podataka	Definicija	Opis
id	bigint	Primary key, Not NULL	identifikacijski ključ
post_id	bigint	Foreign key, Not NULL	vanjski ključ
image	character varying		fotografija prostora

Tabela 5.3 Tablica images

Relacije

Relacijska shema baze podataka sastoji se od skupa povezanih tablica. Radi se o modelu gdje se podaci spremaju u različite tablice bez redundancije. U kontekstu relacijske baze, veza između dvije tablice ostvaruje se vrijednostima primarnog ključa koje se spremaju u vanjski ključ povezane tablice. Primarni ključ unutar tablice se ne smije ponavljati, dok vanjski ključ može. Ovdje nastaje relacija roditelj-dijete gdje roditelj može imati niti jedno ili više djece, dok dijete mora imati roditelja (Baze podataka 2019).



Slika 5.5 Relacija

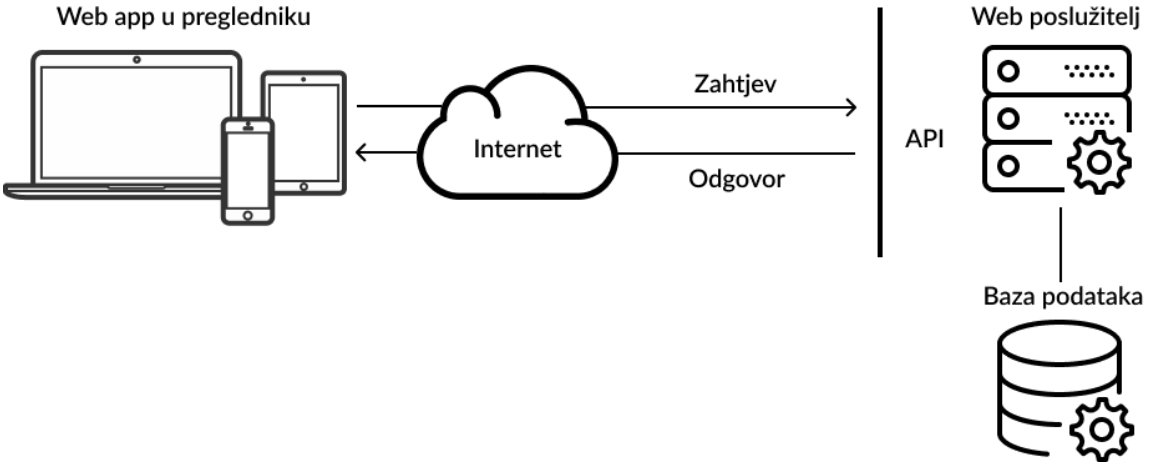
5.2.3. REST API

REST je skraćenica za reprezentativni prijenos stanja (eng. representational state transfer), definira skup arhitektonskih principa pomoću kojih se dizajniraju web usluge gdje se prenose resursi i njihova stanja preko HTTP protokola.

REST web servisi slijede četiri osnovna načela dizajna, a to su:

- korištenje HTTP protokola,
- dizajn bez stanja (eng. stateless),
- struktura direktorija izložena URL-ovima,
- prijenos JSON-a (eng. JavaScript object notation) (Rodriguez 2008).

RESTful API omogućuje izvođenje poziva sa klijentske strane na server i vraća nazad podatke koriteći HTTP protokol. Podaci se vraćaju u JSON formatu gdje su podaci strukturirani i organizirani po principu parova ključ i vrijednost (eng. key - value). Poziv na server može sadržavati i parametre koji specificiraju podatke koji se dobivaju kroz API. Riječ je o servisima koji omogućuju jednostavno izvršavanje osnovnih CRUD operacija za stvaranje (eng. create), čitanje (eng. read), ažuriranje (eng. update) i brisanje (eng. delete). Obavljanje operacija omogućuju HTTP metode poziva GET za dohvaćanje, POST za pisanje podataka, PUT za ažuriranje i DELETE za brisanje. Jednostavno se putem servera vrši komunikacija između klijenta i baze podataka na koju se šalju SQL upiti s odgovarajućim parametrima.



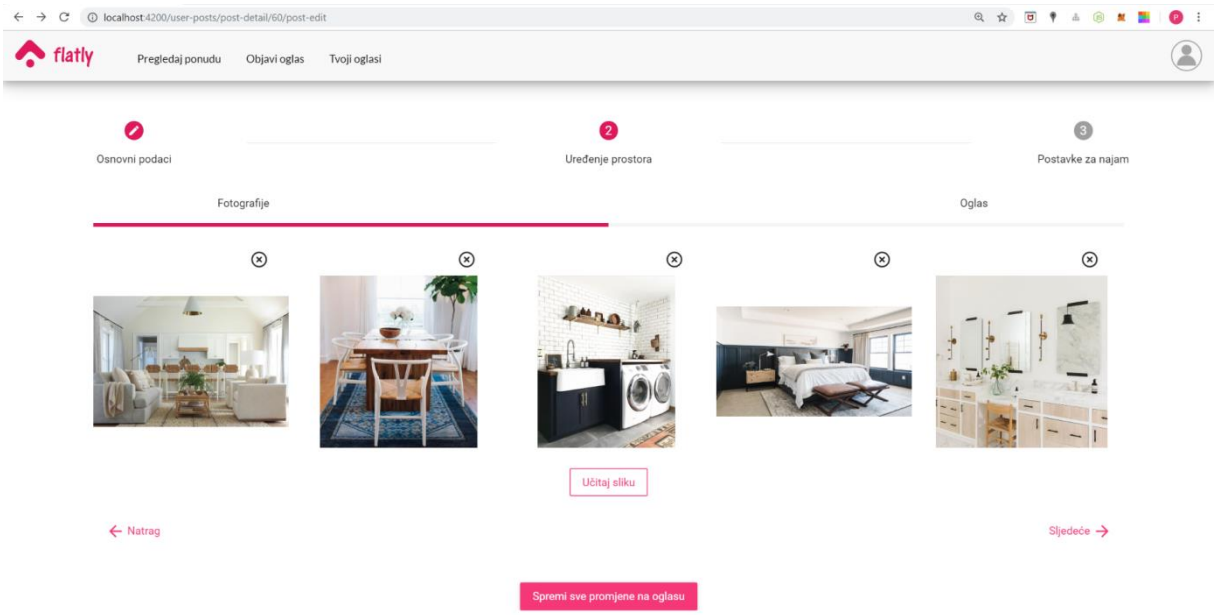
Slika 5.6 RESTful API

5.2.4. Integracija

Angular je front-end framework koji sam po sebi služi za prikaz podataka i obrazaca korisniku, dakle on služi samo kao prednji dio aplikacije koji je uglavnom vizualan. Kako bi aplikacija bila potpuno razvijena i funkcionalna potrebno je Angular integrirati s tehnologijama koje omogućavaju razvoj RESTful API-a koji će vršiti komunikaciju pohranom i čitanjem podataka u za to predviđenu bazu podataka.

U aplikaciji glavnu funkcionalnost predstavlja unos i prikaz oglasa iznajmljivih prostora. U tu svrhu prikazat će se integracija odabranih tehnologija za razvoj te način ostvarenja pojedine funkcionalnosti.

Ključna funkcionalnost aplikacije za koju je potrebno dizajnirati posebno dobro korisničko iskustvo jest stvaranje novog oglasa. Korisnik mora unijeti podatke vezane uz prostor popunjavajući odgovarajući obrazac koji se prikazuje na stranici. Radi se o obrascu s više koraka i smješten je u komponenti „form“. S obzirom na to da je obrazac vrlo velik, u svrhu prikaza integracije, koriste se samo isječci koda koji su relevantni. Za primjer se koristi korak „Uređenje prostora“ gdje je potrebno učitati fotografije prostora te zadati naslov i opis oglasa.



Slika 5.7 Obrazac uređenja prostora

Isječak HTML koda:

```
<mat-step label="Uređenje prostora">
      <mat-horizontal-stepper id="inner-stepper">
        <mat-step label="Fotografije">
          <mat-progress-bar
            [color]="accent"
            [mode]="determinate"
            [value]="50"
            [bufferValue]="75"
          >
          </mat-progress-bar>

          <form class="" [formGroup]="postsForm">
            <div class="grid">
              <div style="position: relative;" *ngFor="let image of images">
                <button
                  mat-button
                  style="position: absolute; right: 0;"
                  (click)="deleteImage(image)"
                >
                  <mat-icon>highlight_off</mat-icon>
                </button>
                <div
                  style="margin-top: 15%; width: 250px; height: 220px;
position: relative;"
                >
                  <img [src]="image" class="scale" />
                </div>
              </div>
            </div>
            <div>
              <input
                type="file"
                accept="image/*"
                style="display: none;"
                #image
                name="image"
                (change)="onFileSelected($event.target.files)"
              />

              <button
                mat-stroked-button
                color="accent"
                style="display: block; margin: auto; margin-top: 2%; margin-
bottom: 2%;"
                (click)="image.click()"
              >
```

```
            Učitaj sliku
          </button>
        </div>

        <div style="position: relative;">
          <button mat-button color="accent" matStepperPrevious>
            <mat-icon>arrow_back</mat-icon>
            Natrag
          </button>
          <button
            style="position: absolute; right: 0;"
            mat-button
            color="accent"
            matStepperNext
          >
            Sljedeće
            <mat-icon>arrow_forward</mat-icon>
          </button>
        </div>
      </form>
    </mat-step>

    <mat-step label="Oglas">
      <mat-progress-bar
        [color]="accent"
        [mode]="determinate"
        [value]="100"
        [bufferValue]="75"
      >
      </mat-progress-bar>

      <form class="container" [formGroup]="postsForm">
        <mat-form-field color="accent" appearance="outline">
          <mat-label>Odredite naziv oglasa</mat-label>
          <input
            matInput
            placeholder="Naslov"
            formControlName="title"
          />
        </mat-form-field>
        <mat-form-field color="accent" appearance="outline">
          <mat-label>Uredite opis</mat-label>
          <input
            matInput
            placeholder="Opis"
            formControlName="description"
          />
        </mat-form-field>
```

```
            <div>
              <button mat-button color="accent" matStepperPrevious>
                Natrag
              </button>
              <button mat-raised-button color="accent" matStepperNext>
                Sljedeće
              </button>
            </div>
          </form>
        </mat-step>
      </mat-horizontal-stepper>
    </mat-step>
```

form.component.html

Ključne dijelove koda predstavljaju direktive koje Angular pruža.

```
<form class="container" [formGroup]="postsForm">
```

Angular direktiva „FormGroup“ injektira se u stvorenu komponentu i ona označava obrazac kojega element mora gledati. Obrazac se definira u kontroler dijelu komponente prilikom inicijalizacije iste.

```
constructor(private postService: PostService, private fb: FormBuilder) {}

  ngOnInit() {
    this.postsForm = this.fb.group({
      status: [false],
      title: [],
      description: [],
      bills_included: [false],
      country: [],
      city: [],
      address: [],
      price: [],
      squares: [],
      type: [],
      available_date: [],
      walkout_date: [],
      furnished: [false],
      bed: [],
      room: [],
      pet: [false],
      parking: [false],
      wifi: [false]
    });
  }
```

form.component.ts

Prilikom učitavanja slika, iste se spremaju u polje string-ova. Fotografije s obrascem oglasa spremaju se u instancu klase „Posts“.

```
export class Posts {
  id: number;
  user_id: number;
  status: boolean;
  title: string;
  description: string;
  bills_included: boolean;
  country: string;
  city: string;
  address: string;
  price: number;
  squares: number;
  type: string;
  available_date: Date;
  walkout_date: Date;
  furnished: boolean;
  bed: number;
  room: number;
  pet: boolean;
  parking: boolean;
  wifi: boolean;
  image: string;
  images: string[];
  deletedImages: any[];
}
```

posts.ts

Uneseni podaci se spremaju u instancu „posts“ koja se šalje na server koristeći metodu „addPost“ implementiranu unutar injektiranog servisa „PostService“.

```
onSubmit() {
    this.submitted = true;

    if (this.postsForm.invalid) {
      return;
    }

    this.posts = this.postsForm.value;
    this.posts.images = this.images;

    this.save();
  }

  private save() {
```

```
    this.postService.addPost(this.posts).subscribe();
  }
```
form.component.ts

Metoda „addPost“ šalje instancu klase „Posts“ na određenu rutu servera gdje će se obaviti POST request na bazu.

```
addPost(post: Posts): Observable<any> {
    return this.http.post<any>(`/api/form`, post).pipe(
      map(res => {
        return res.rows;
      })
    );
  }
```
post.service.ts

POST request otvara novu transakciju gdje se kroz „req.body“ dobiva tijelo zahtjeva, odnosno uneseni parametri koji se predaju SQL upitu „insertPost“ i „insertImages“.

```
router.post("/", jsonParser, (req, res, next) => {
  const user_id = req.cookies["user_id"];

  transaction
    .getPool()
    .connect()
    .then(client => {
      client
        .query(queries.begin)
        .then(() => {
          const sql = queries.insertPost;
          let params = [
            user_id,
            false,
            req.body.title,
            req.body.description,
            req.body.bills_included,
            req.body.country,
            req.body.city,
            req.body.address,
            req.body.price,
            req.body.squares,
            req.body.type,
            req.body.furnished,
            req.body.bed,
            req.body.room,
            req.body.pet,
            req.body.parking,
```

```
            req.body.wifi
          ];

          return client.query(sql, params);
        })
        .then(results => {
          let post_id = results.rows[0].id;

          const sql2 = queries.insertImages;
          let arrayImages = [];
          req.body.images.forEach(img => {
            arrayImages.push([post_id, img]);
          });
          let sql = formatSQL(sql2, arrayImages);

          return client.query(sql);
        })
        .then(results2 => {
          res.status(200).json(results2);
          transaction.commit(client);
        })
        .catch(err => {
          console.log("error", err);
          client.query(queries.end);
          transaction.rollback(client);
        })
        .then(() => {
          client.release();
        });
    })
    .catch(err => {
      console.log("error", err);
      res.status(401).json(err);
    });
});
```

form.js

SQL upit „insertPost“ prima sve parametre poslane s klijenta i unose se u tablicu „posts“. Jednom kada je redak unesen upit vraća identifikacijski ključ retka koji se prosljeđuje za potrebe spremanja fotografija prostora. Svaka fotografija se sprema kao zasebni redak u tablici „images“ koja sadržava vanjski ključ na tablicu „posts“ kako bi se znalo kojem oglasu pripadaju fotografije.

```
insertPost: `INSERT INTO posts
              ("user_id", "status", "title", "description", "bills_included",
"country", "city", "address", "price",
```

```
                "squares", "type", "furnished", "bed", "room", "pet", "parking",
"wifi")
                VALUES
                ($1, $2, $3, $4, $5, $6, $7, $8, $9,
                $10, $11, $12, $13, $14, $15, $16)
                RETURNING id`
```

queries.js

SQL upit „insertImages“ prima polje objekta koji se sastoji od atributa „id“ oglasa i string fotografije.

```
insertImages: `INSERT INTO images
                    ("post_id", "image")
                    VALUES
                    %L`
```

queries.js

Tako korisnik unosi novi redak u bazu podataka, dok pregled pojedinog oglasa zahtjeva HTTP GET upit na bazu. Korisnik odabire oglas kojega želi vizualizirati odabirom jedne od kartice koje su prikazane na pregledu oglasa. Prilikom klika na karticu, uzima se identifikacijski ključ oglasa i postavlja se kao parametar URL-a koji definira rutu.

```
ngOnInit(): void {
    this.id = +this.route.snapshot.paramMap.get("id");

    this.postService.getPost(this.id).subscribe(post => {
      this.post = post;

      for (let im = 0; im < post.images.length; im++) {
        this.tiles[im].text = (<any>post.images[im]).image;
      }
    });
  }
```

post-detail.component.ts

Iz URL-a dobiva se parametar „id“ koji se šalje injektiranom servisu „PostService“ koji implementira metodu „getPost“.

```
getPost(id: number): Observable<Posts> {
    return this.http.get<any>(`/api/view/post-detail/${id}`).pipe(
      map(res => {
        return res.rows[0];
      })
    );
```

```
  }
```
post.service.ts

Metoda „get“ šalje parametar na određenu rutu servera te jednom kada dobije rezultat vratit će odgovor servera na odgovarajuću stranicu prikaza.

```
router.get("/:id", (req, res) => {
  const client = new Client();
  let posts;
  client
    .connect()
    .then(() => {
      const sql = queries.getPostDetails;
      let params = [req.params.id];
      return client.query(sql, params);
    })
    .then(post => {
      posts = post;
      const sql = queries.getImages;
      let params = [req.params.id];
      return client.query(sql, params);
    })
    .then(images => {
      posts.rows[0].images = images.rows;
      res.status(200).json(posts);
    })
    .catch(err => {
      console.log("error", err);
    })
    .then(() => client.end());
});
```
post-detail.js

Iz baze se dohvaća odgovarajući oglas sa svim fotografijama tog oglasa.

```
getPostDetails: `SELECT
                    *
                  FROM posts, users
                  WHERE posts.id = $1
                  AND posts.user_id = users.id`,

getImages: `SELECT * FROM images WHERE post_id = $1`
```
queries.js

Odgovor baze se kroz server vraća na servis metodu „getPost“ ranije prikazanu gdje se rezultat vraća na prikaz komponente.

```
<mat-grid-list cols="4" rowHeight="150px" role="list">
  <mat-grid-tile
    *ngFor="let tile of tiles"
    [colspan]="tile.cols"
    [rowspan]="tile.rows"
    [style.background]="tile.color"
  >
    <div>
      <img
        role="listitem"
        mat-card-image
        class="zoom"
        [src]="tile.text"
        alt="Slika prostora"
      />
    </div>
  </mat-grid-tile>
</mat-grid-list>

<div class="grid center" *ngIf="post">
  <div>
  </div>
  <div class="detail">
    <div class="title" style="margin-bottom:5%;">
      <h2>
        {{ post.title }}
      </h2>
      <h2 style="text-align: end;">{{ post.price }}</h2>
      <h3>{{ post.type }}, {{ post.squares }} m2</h3>
    </div>

    <p class="span-col-2" style="margin-bottom:5%;">
      {{ post.description }}
    </p>

    <div class="bool">
      <div>
        <div class="center anemities">
          <img
            src="assets/images/maps-and-flags.svg"
            alt="Lokacija"
            class="anemities-icon"
          />
          <div class="center anemities">
            <p class="anemities">{{ post.city }}, {{ post.address }}</p>
          </div>
        </div>

        <div style="margin-bottom:3%;"></div>
```

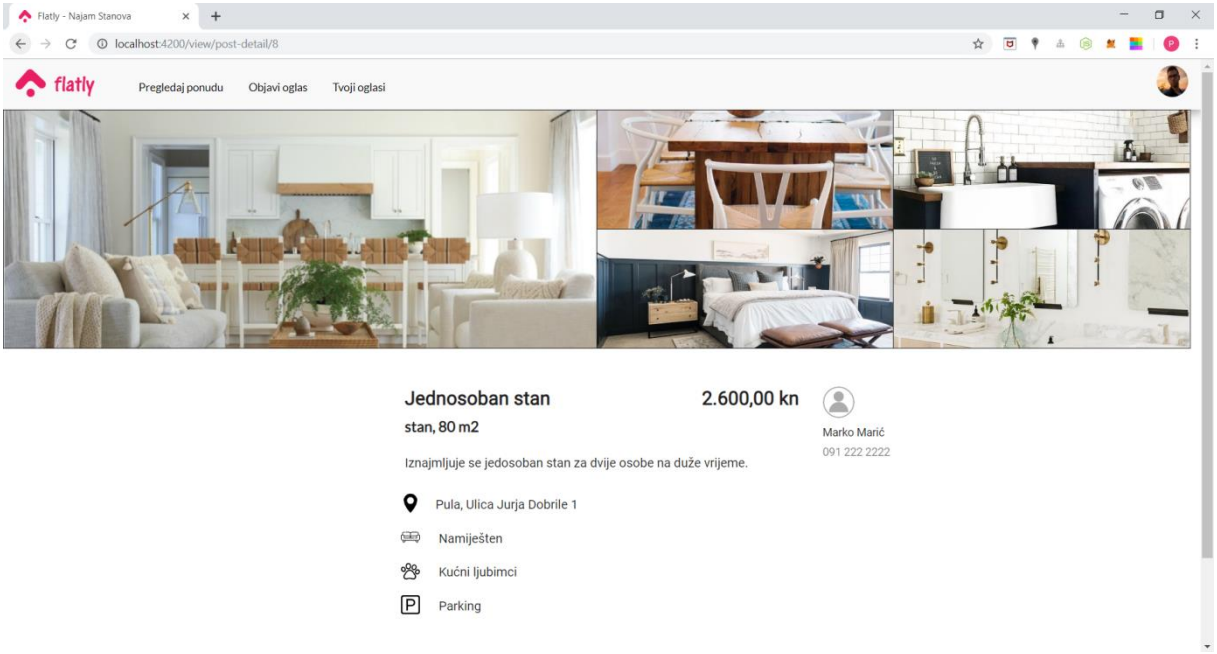
```
<div *ngIf="post.bills_included">
  <div class="center anemities">
    <img
      src="assets/images/bill.svg"
      alt="Rezije"
      class="anemities-icon"
    />
  </div>
  <div class="center anemities">
    <p class="anemities">
      Uključene režije
    </p>
  </div>
</div>

<div style="margin-bottom:3%;"></div>

<div *ngIf="post.furnished">
  <div class="center anemities">
    <img
      src="assets/images/couch.svg"
      alt="Namijesten stan"
      class="anemities-icon"
    />
  </div>
  <div class="center anemities">
    <p class="anemities">
      Namiješten
    </p>
  </div>
</div>

<div style="margin-bottom:3%;"></div>

<div *ngIf="post.pet">
  <div class="center anemities">
    <img
      src="assets/images/pawprint.svg"
      alt="Zivotinje"
      class="anemities-icon"
    />
  </div>
  <div class="center anemities">
    <p class="anemities">
      Kućni ljubimci
    </p>
  </div>
</div>
```

```
        <div style="margin-bottom:3%;"></div>

        <div *ngIf="post.parking">
          <div class="center anemities">
            <img
              src="assets/images/parking.svg"
              alt="Parking"
              class="anemities-icon"
            />
          </div>
          <div class="center anemities">
            <p class="anemities">
              Parking
            </p>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div *ngIf="isLoggedIn">
    <mat-card class="user">
      <mat-card-header class="user-grid center">
        <div mat-card-avatar class="header-image"></div>
        <mat-card-title
          >
          {{ post.name }} {{ post.surname }}</mat-card-title
        >
        <mat-card-subtitle>
          {{ post.phone }}</mat-card-subtitle
        >
      </mat-card-header>
    </mat-card>
  </div>

</div>
```

post-detail.component.html

Angular direktiva “ngFor“ služi za iteriranje polja „images“ te prikaz pojedine fotografije u galeriji. Dok, „ngIf“ služi za ispis odgovarajućih atributa za koje objekt sadrži vrijednost.

Korisnik tako dobiva detaljan pregled odabranog oglasa sa svim relevantnim specifikacijama kao što je prikazano na slici u nastavku.

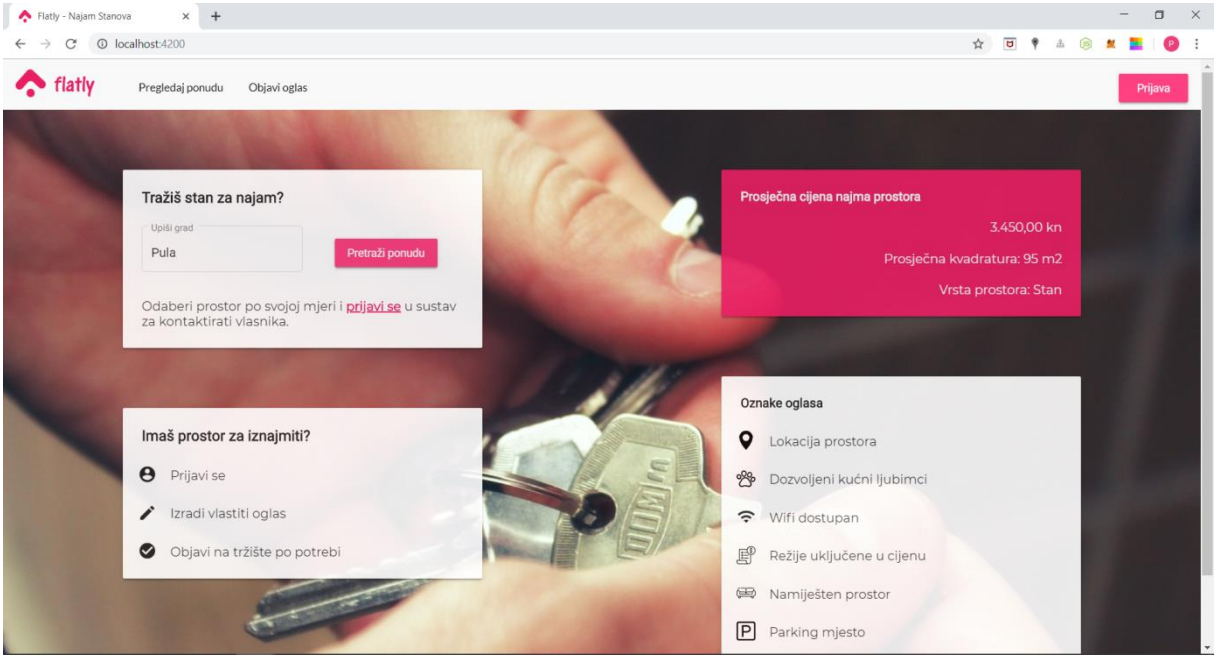


Slika 5.8 Prikaz detalja odabranog oglasa

6. Korisničke upute

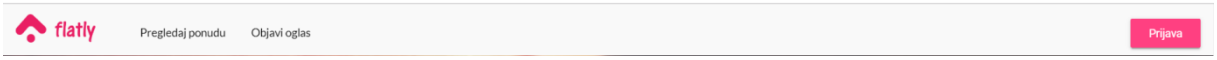
Način korištenja aplikacije prikazan je kroz snimke zaslona (eng. screenshot) sa stvarnog sučelja aplikacije.

Prilikom dolaska na web mjesto gdje se nalazi aplikacija putem web preglednika, korisnik dolazi na odredišnu stranicu (eng. landing page) koja mu daje do znanja o kakvoj je aplikaciji riječ te koje mogućnosti može koristiti.



Slika 6.1 Odredišna stranica

Aplikacija ima jednu glavnu alatnu traku (eng. toolbar) koja se nalazi na vrhu ekrana. Alatna traka može imati više stanja koja ovise o statusu prijave korisnika. Ako korisnik nije prijavljen, alatna traka pokazuje mogućnost "Pregledaj ponudu" koja vodi na ekran koji prikazuje aktivne oglase, "Objavi oglas" koji u ovom slučaju neće odvesti na odgovarajući obrazac, već na ekran prijave ili registracije korisnika jer korisnik ne može objavljivati oglase bez da se prije toga prijavio na vlastiti račun. Dok, u desnom kutu, alatna traka daje opciju "Prijava" koja vodi korisnika na ekran za prijavu ili registraciju novog korisnika kao što je prikazano na slici u nastavku.

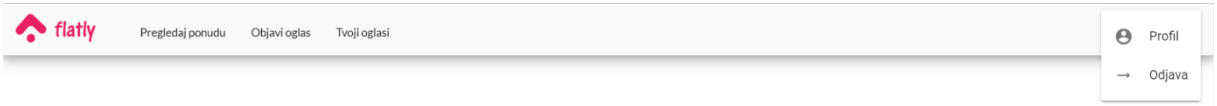


Slika 6.2 Alatna traka prije prijave

Kada bi se korisnik prijavio na vlastiti račun ili napravio novi račun ako ga već ne posjeduje, alatna traka bi se osvježila s novim ponudama. Opcija "Objavi oglas" vodila bi na obrazac za unos novog oglasa, dok bi nova opcija "Tvoji oglasi" vodila na ekran gdje su prikazani svi oglasi u vlasništvu korisnika i njihovo stanje.

Slika 6.3 Alatna traka poslije prijave

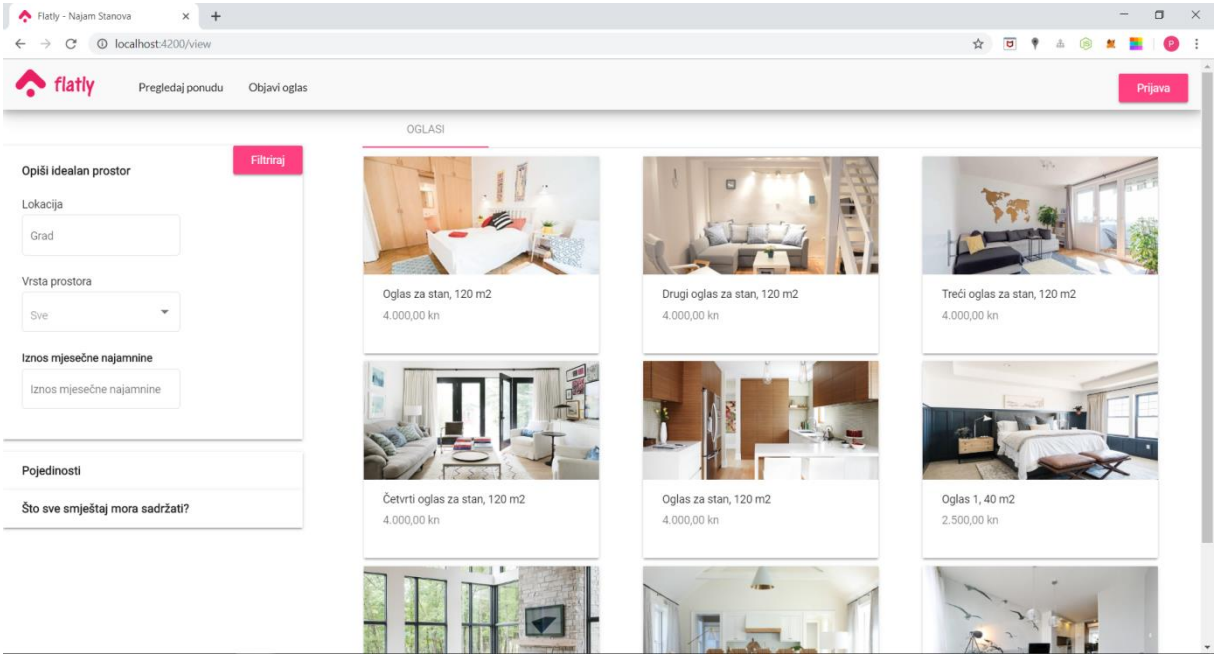
Ikonica u desnom kutu alatne trake predstavlja sliku profila korisnika, te klikom na ikonicu otvara se izbornik koji daje dvije opcije. Opcija "Profil" vodi korisnika na ekran gdje korisnik može upravljati svojim osobnim podacima, dok opcija "Odjavi" odjavljuje korisnika s vlastitog računa i vraća aplikaciju na prvobitno stanje.



Slika 6.4 Izbornik korisnika

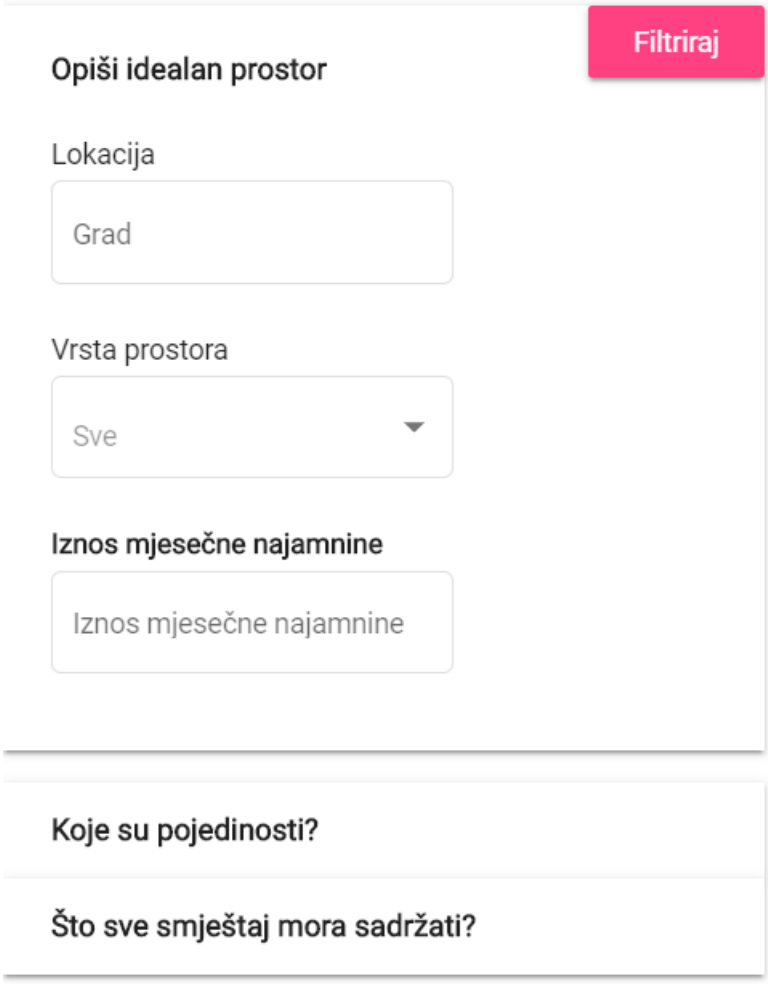
Kako bi se najbolje prikazao tok aplikacije moguće je simulirati rad fiktivnog korisnika koji po prvi puta koristi aplikaciju. Nakon što je korisnik sletio na odredišnu stranicu s izbornika će odabrati opciju "Pregledaj ponudu" kako bi se uvjerio da je ova aplikacija namijenjena upravo onome što je njemu potrebno, te kako bi dobio uvid u tržište i strukturu oglasa. U ovom slučaju korisnik još nije prijavljen niti registriran.

Ekran "Pregledaj ponudu" prikazuje sve oglase koji su trenutno aktivni na tržištu i podijeljen je na dva dijela. S lijeve strane nalazi se filter kojega korisnik može koristiti kako bi efikasnije pretraživao oglase koji su za njega zanimljivi, dok se s desne strane nalaze kartice koje sadrže istaknute govoreće podatke vezane uz iznajmivu nekretninu.

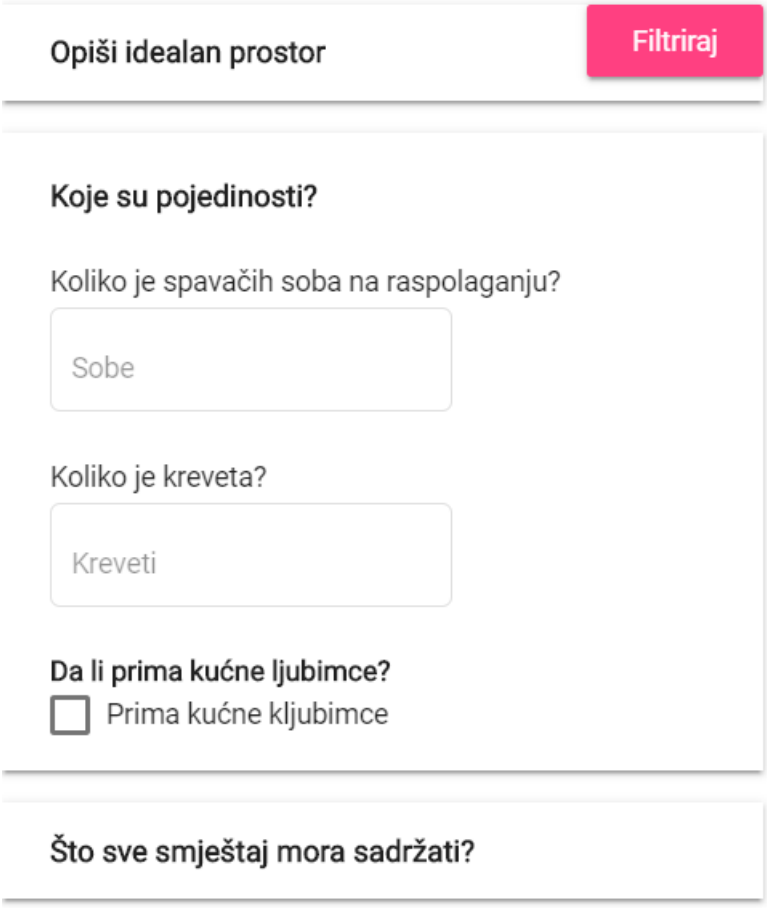


Slika 6.5 Pregled ponude

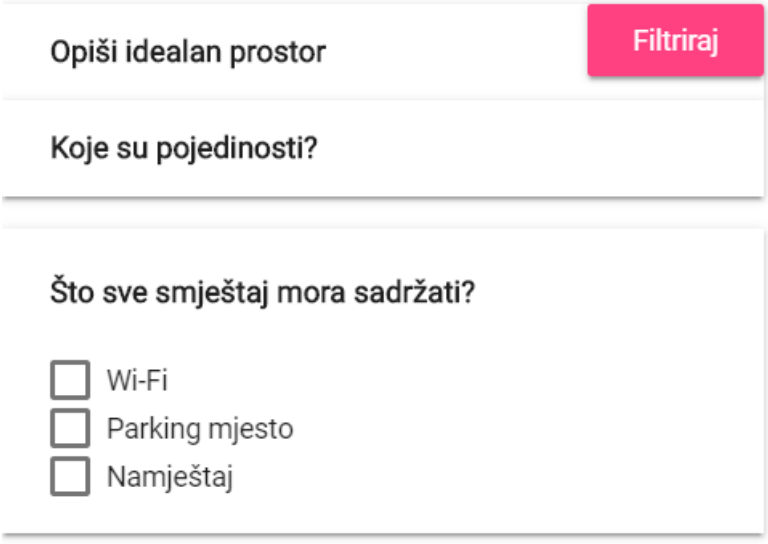
Na lijevoj strani ekrana nalazi se filter kojega korisnik može koristiti kako bi pretraživao oglase po specifičnim parametrima. Parametra za pretraživanje može biti jako puno pa se u tu svrhu koristi komponenta s ekspanzijom i podijeljena je na tri dijela: „Opiši idealan prostor“, „Koje su pojedinosti?“ i „Što sve smještaj mora sadržati?“. Korisnik unosi svoje preferencije za pretraživanje te klikom na gumb "Filtriraj" dobiva rezultate pretraživanja. Dijelovi filtera prikazani su u sljedećem nizu snimaka zaslona.



Slika 6.6 Filter opis prostora

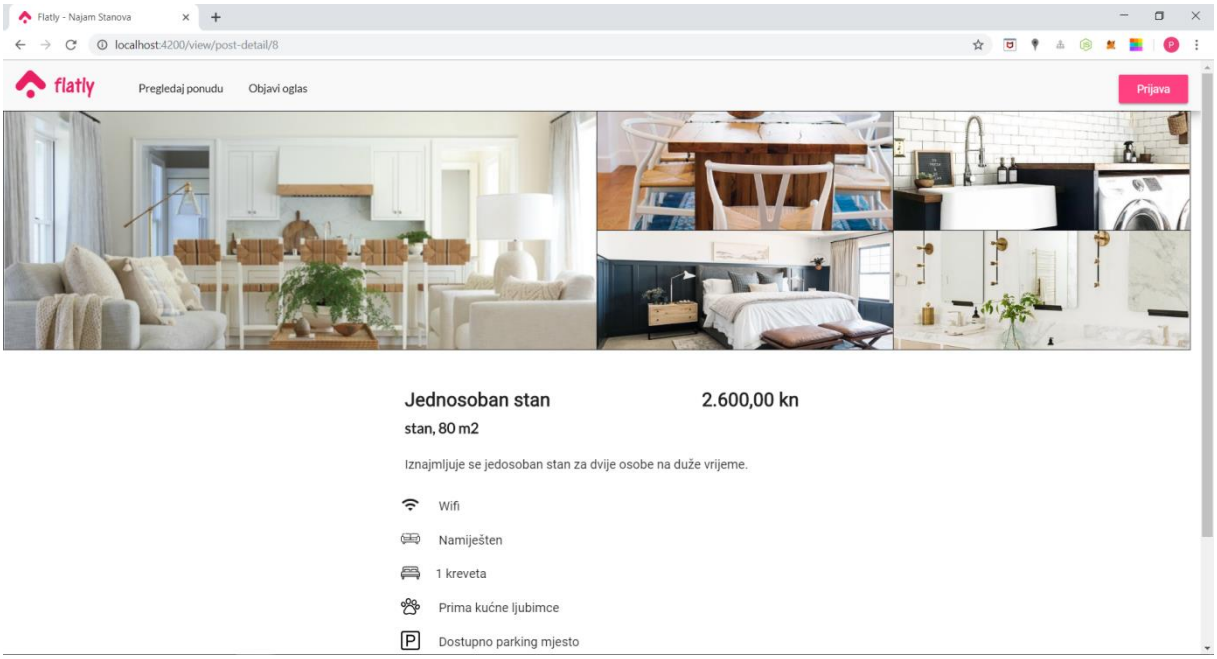


Slika 6.7 Filter pojedinosti



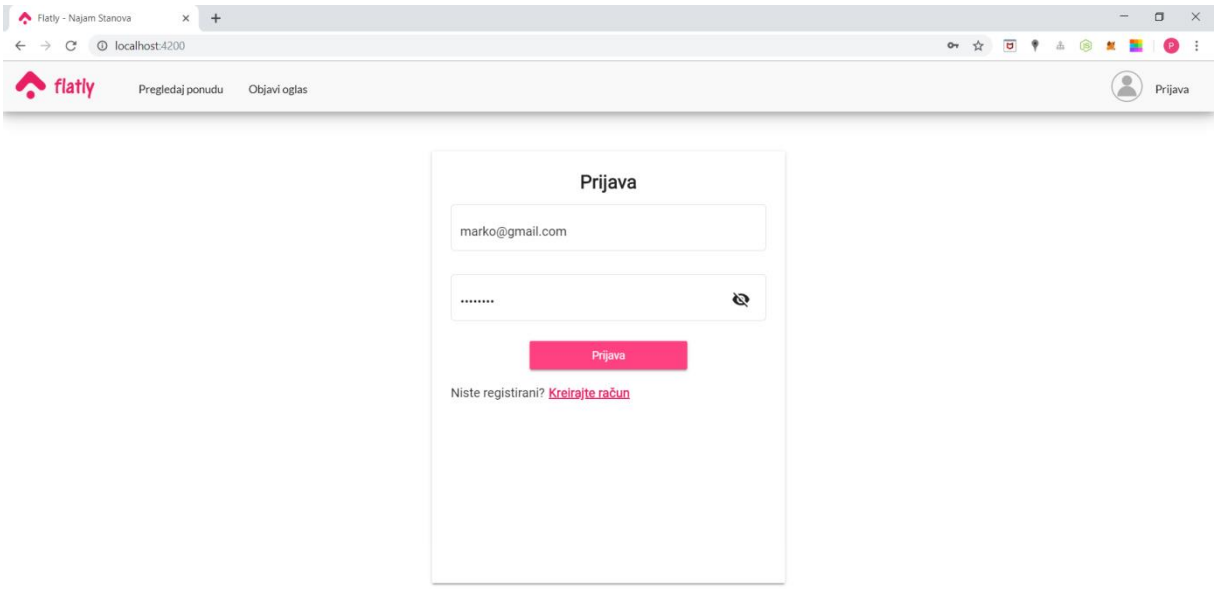
Slika 6.8 Filter sadržaja

Korisnik može kliknuti na jednu karticu što će ga odvesti na ekran koji detaljno prikazuje odabrani oglas. No, ako korisnik nije prijavljen neće mu biti dostupni podaci poput adrese na kojoj se nekretnina nalazi, ime i prezime te kontakt broj telefona vlasnika oglasa.

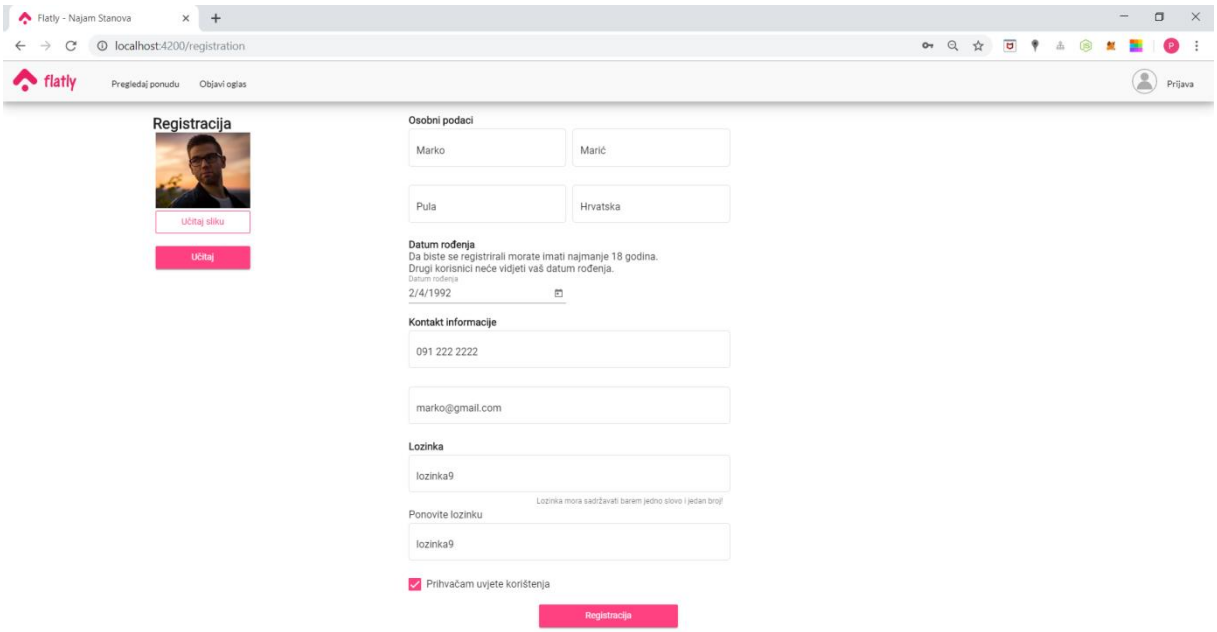


Slika 6.9 Detalji oglasa bez kontakta

Jednom kada korisnik želi stupiti u kontakt s vlasnikom oglasa ili pak želi objaviti vlastiti oglas, potrebno je prijaviti se ili registrirati u sustav. Ekran prijave prikazan je na slici u nastavku. Za prijavu je potrebno unijeti elektroničku poštu i lozinku kako bi se korisnik autentificirao. Ako korisnik ne posjeduje račun, ispod gumba za prijavu postoji inačica "Kreirajte račun" koja vodi do ekrana registracije. Korisnik se može registrirati u sustav sa svojim osobnim i kontakt informacijama te postavljajući lozinku za pristup računu. Ekran registracije korisnika prikazan je na slici u nastavku.

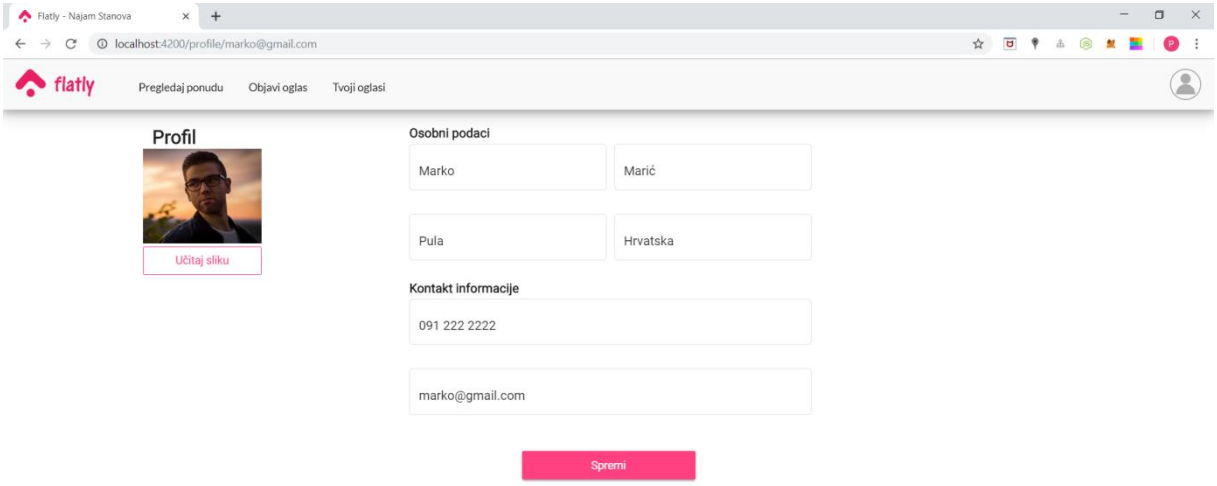


Slika 6.10 Prijava korisnika



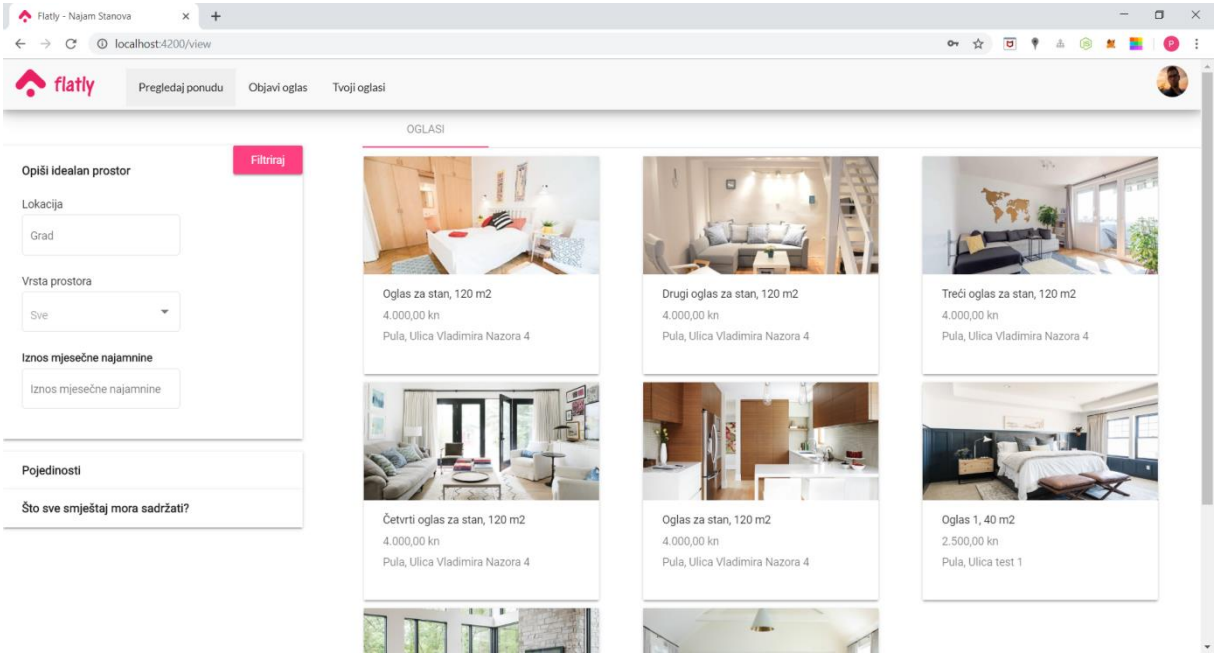
Slika 6.11 Registracija korisnika

Jednom kada je korisnik registriran i prijavljen u sustav, može pogledati svoj profil i ažurirati osobne i kontakt informacije kao što je prikazano na slici u nastavku. Ujedno, može se odjaviti s prijavljenog računa putem korisničkog izbornika.

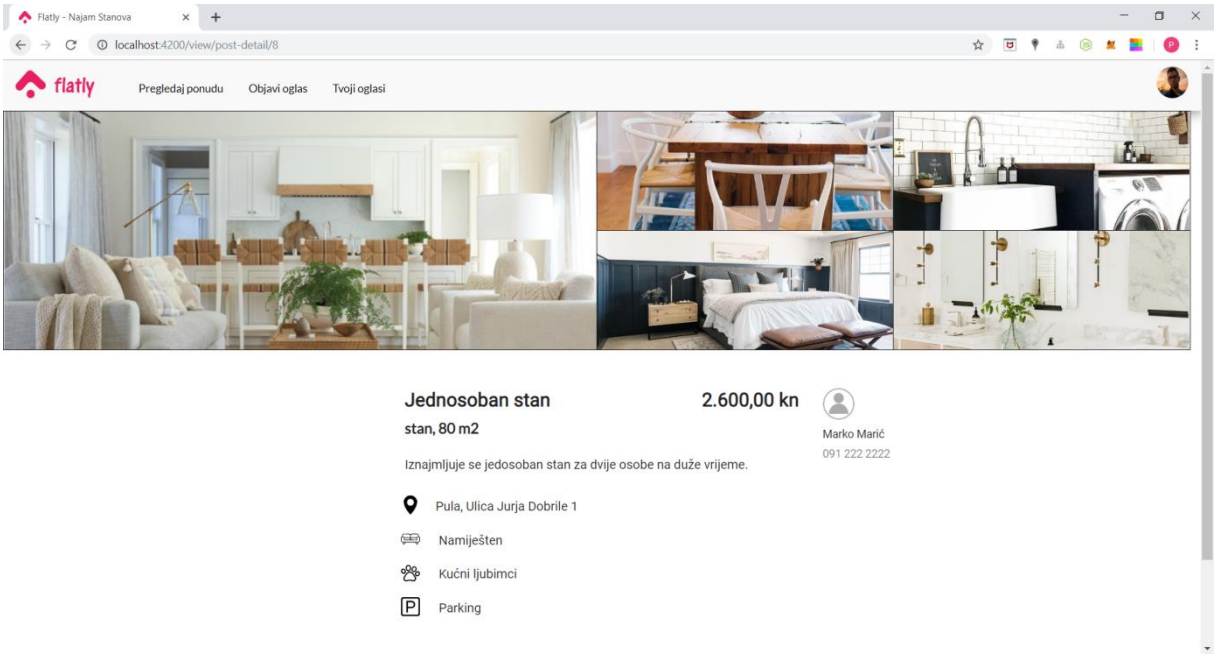


Slika 6.12 Profil korisnika

Nadalje, korisnik može pregledavati ponudu imajući uvid u lokaciju na kojoj se nekretnina nalazi, te prilikom detaljnog pregleda odabranog oglasa pružaju se kontakt podaci vlasnika oglasa s kojim se može sklopiti ugovor o najmu. Ekrani za pregled ponude i detaljan pregled oglasa prikazani su na slici 6.13 i slici 6.14 u nastavku rada.



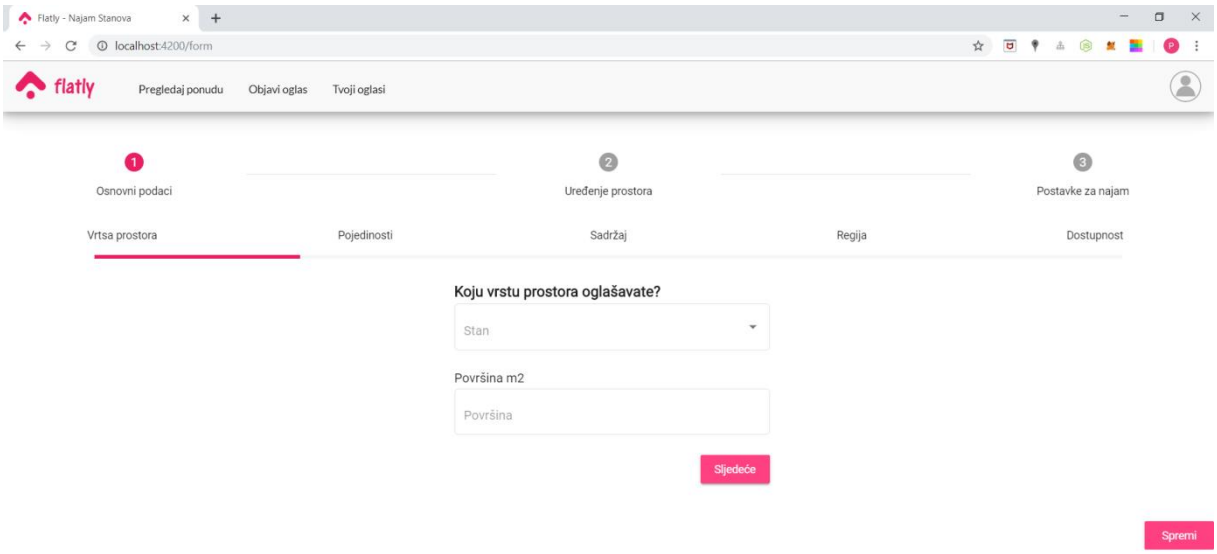
Slika 6.13 Pregled ponude sa lokacijom



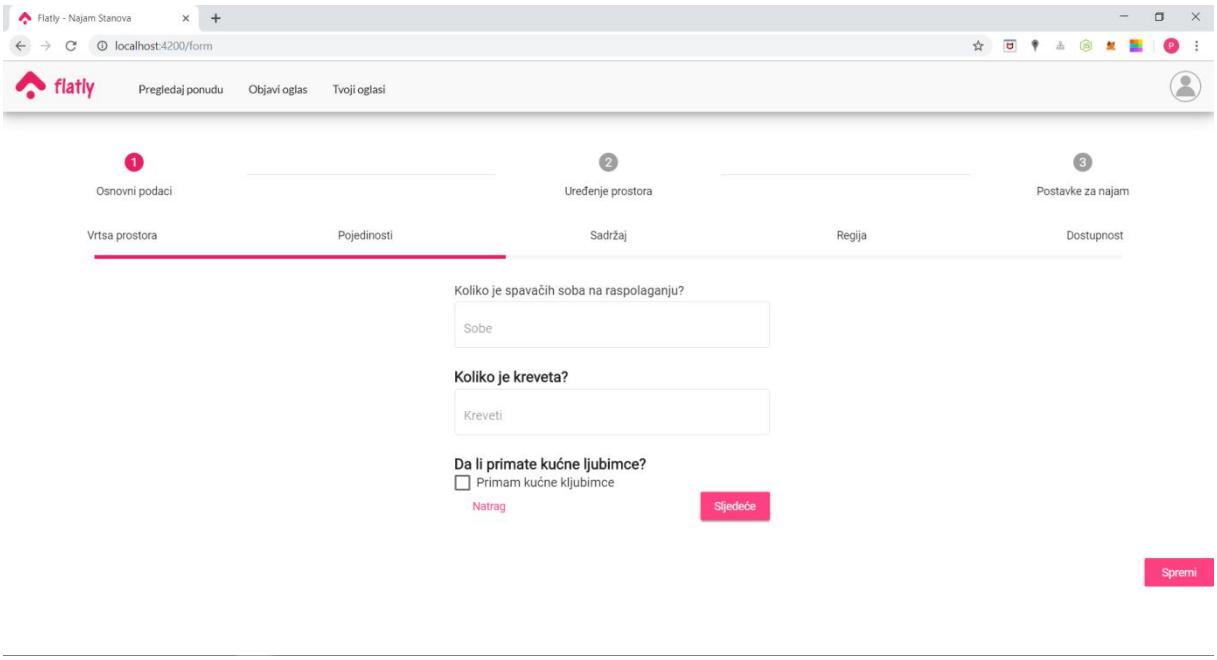
Slika 6.14 Detalji oglasa sa kontaktom

Korisnik koji posjeduje račun u sustavu može kreirati vlastiti oglas odabirom opcije "Objavi oglas" s izbornika. Obrazac za ispunjavanje oglasa potpuno je prilagođen stanodavcima koji žele ostvariti primamljiv oglas pružajući sve specifikacije u koje bi potražnja trebala imati uvid. Obrazac se sastoji od tri glavna koraka, a to su "Osnovni podaci", "Uređenje prostora" i "Postavke za najam".

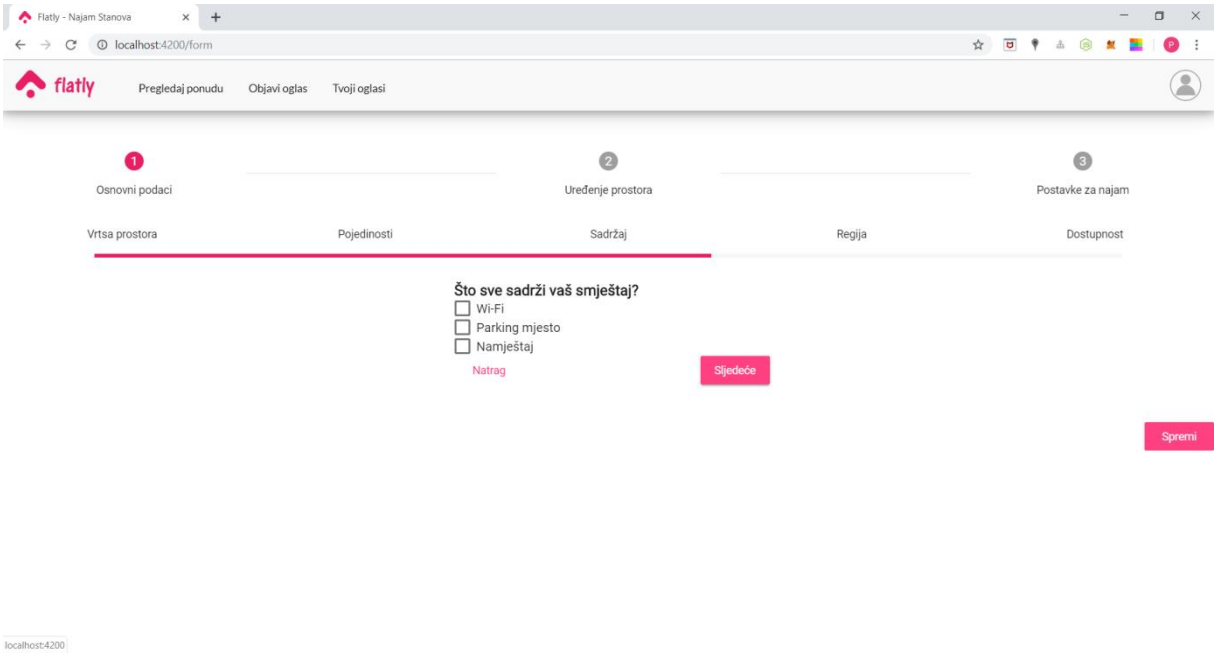
Osnovni podaci sadrže opis vrste prostora koji se izdaje, njegove pojedinosti, sadržaj, lokacija i dostupnost kao što je prikazano u sljedećem nizu snimaka zaslona.



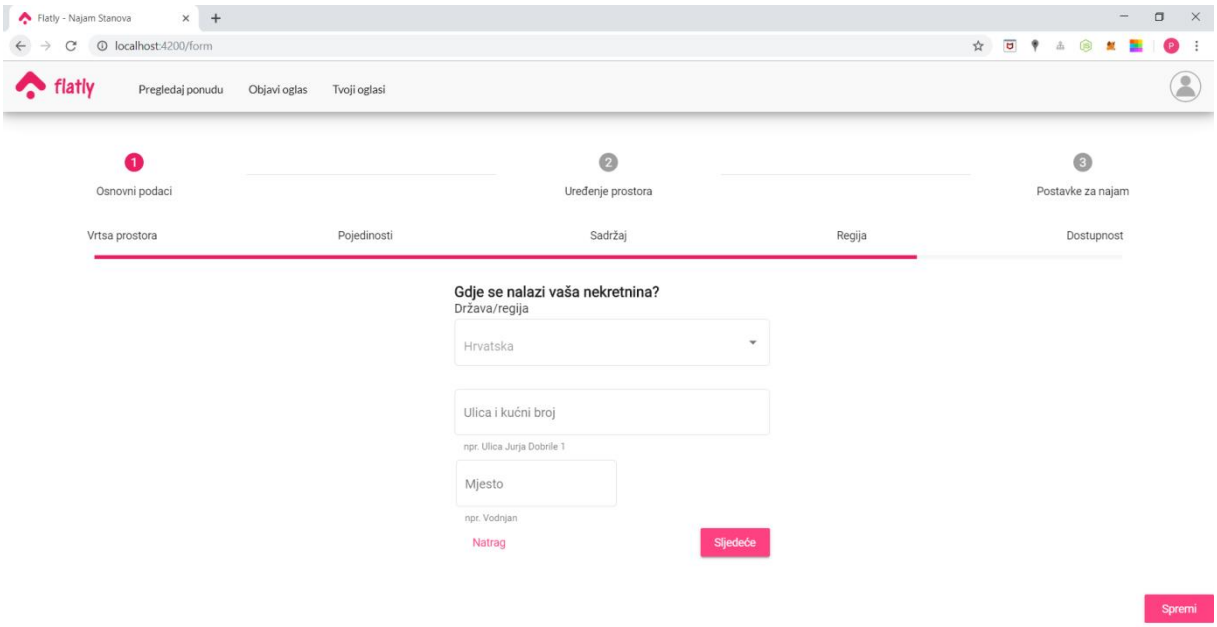
Slika 6.15 Vrsta prostora



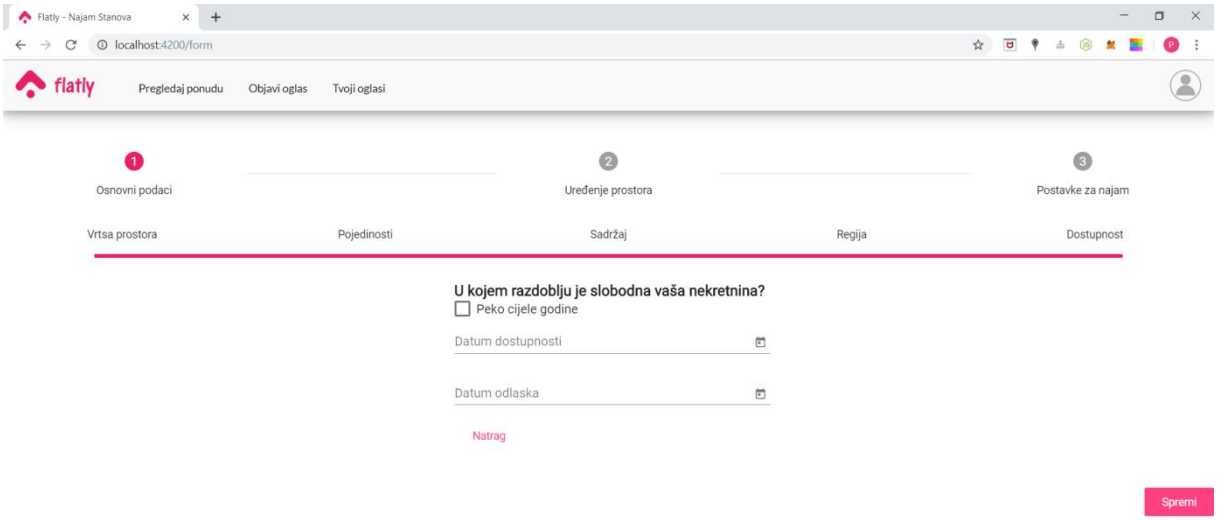
Slika 6.16 Pojedinosti



Slika 6.17 Sadržaj

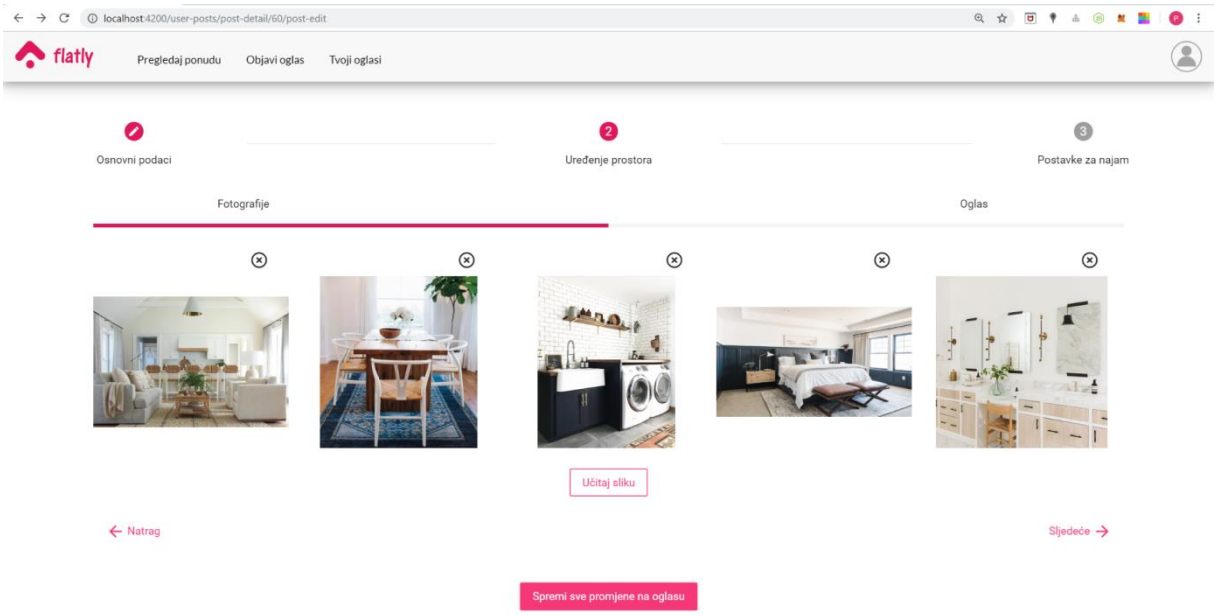


Slika 6.18 Lokacija

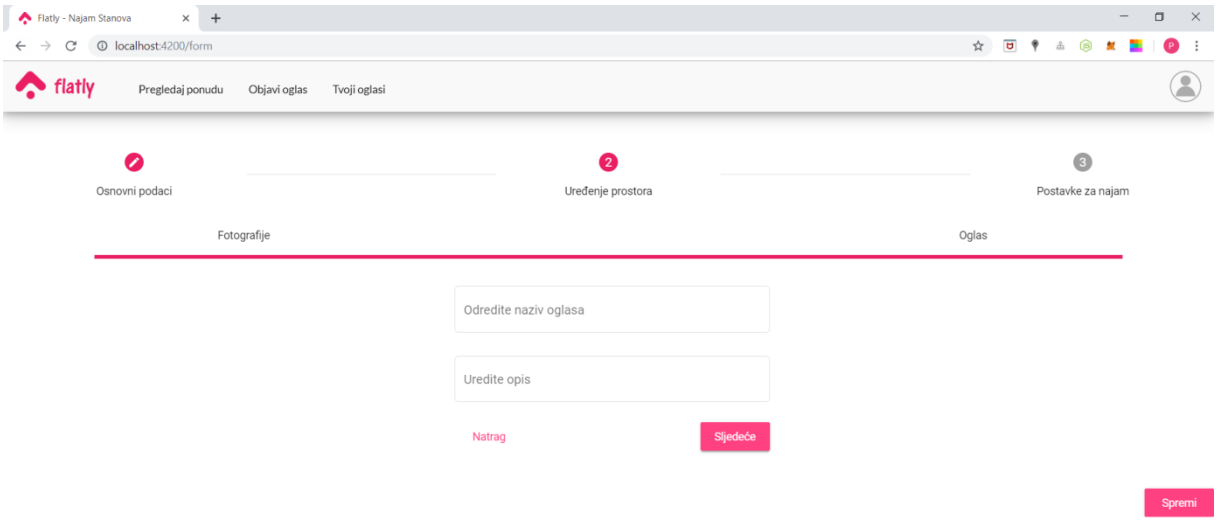


Slika 6.19 Dostupnost

Drugi korak pri stvaranju novoga oglasa jest uređenje prostora što uključuje učitavanje fotografija prostora te slaganje oglasa unoseći naslov i opis nekretnine koja se oglašava.

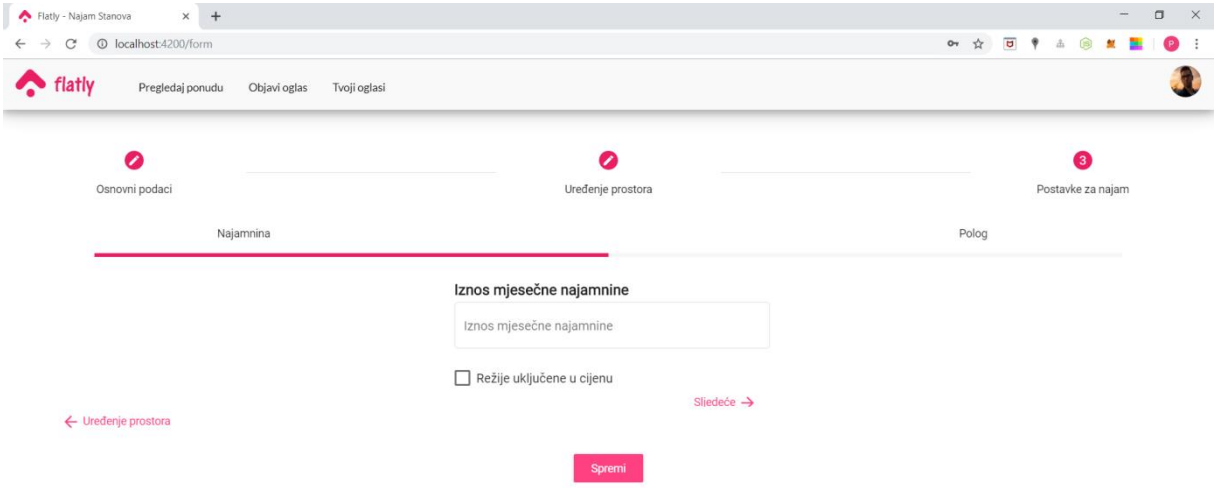


Slika 6.20 Učitavanje fotografija

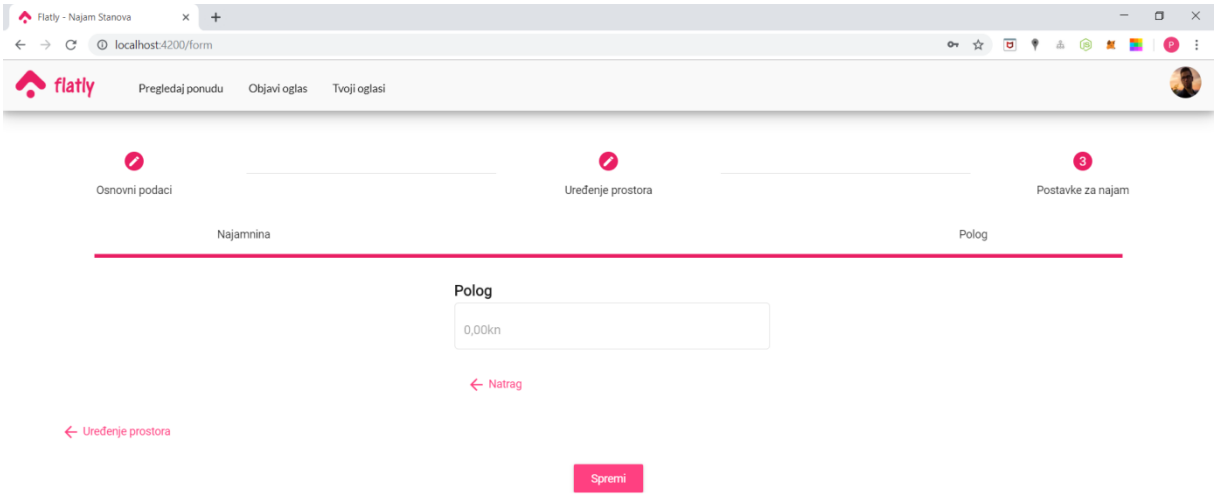


Slika 6.21 Oglas

Treći korak pri stvaranju novoga oglasa jesu postavke za najam koje uključuju unos iznosa mjesečne najamnine te specifikacija da li su režije uključene u cijenu najma. Ako režije nisu uključene, postoji mogućnost specificiranja režija koje je podstanar dužan plaćati te njihov prosječan iznos. Ako postoji potreba za uplaćivanja pologa za ugovaranje stana, potrebno je specificirati iznos u obrascu.

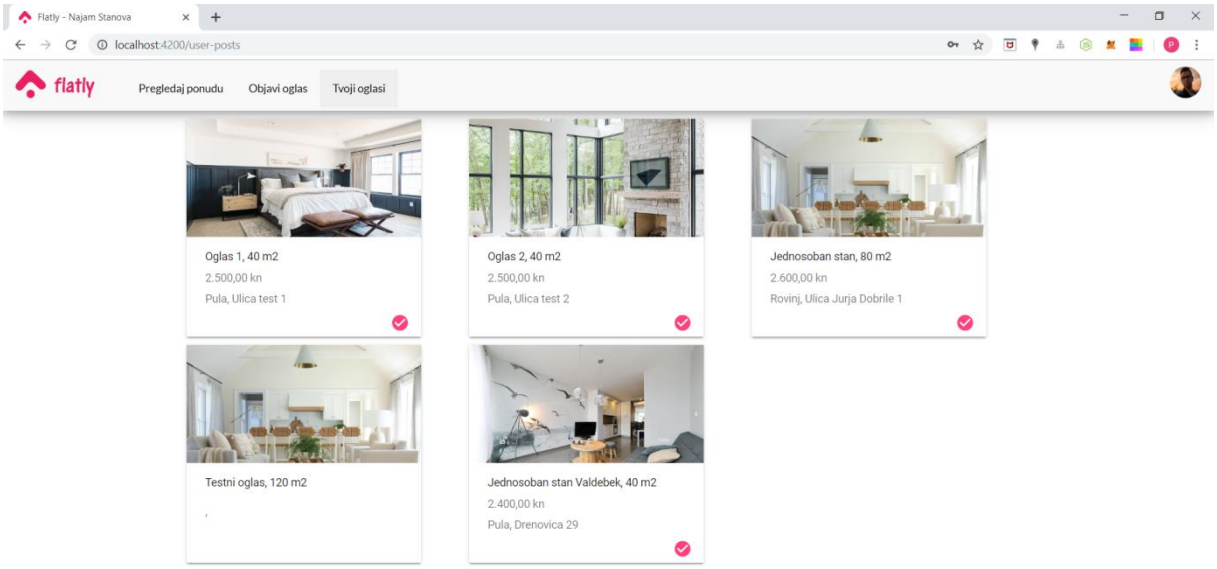


Slika 6.22 Namjamnina



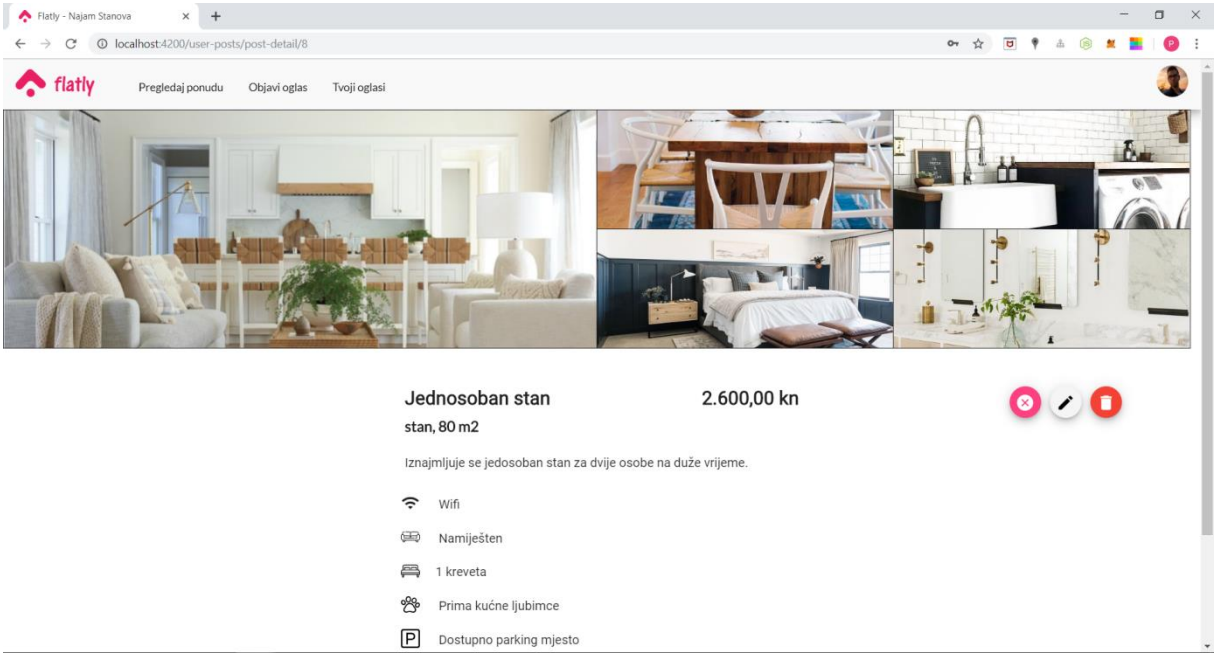
Slika 6.23 Polog

Nakon što je oglas unesen sa strane korisnika kroz obrazac, isti je dostupan na ekranu "Tvoji oglasi" gdje vlasnik može pregledavati svoje oglase i njihov status kao što je prikazano na slici u nastavku.



Slika 6.24 Oglasi korisnika

Klikom na jednu karticu otvara se ekran s detaljima oglasa s alatnom trakom na desnoj strani s tri kružna gumba čijim odabirom korisnik može deaktivirati ili aktivirati oglas kako bi ga postavio na tržište, urediti oglas ako želi promijeniti određene specifikacije te potpuno obrisati oglas ako mu više nije potreban kao što je prikazano na slici u nastavku.



Slika 6.25 Detalji oglasa korisnika

7. Zaključak

Cilj ovog diplomskog rada je razvoj web aplikacije za iznajmljivanje stanova. Aplikacija je razvijena koristeći tehnologije za razvoj prednjeg i stražnjeg ureda kao što su Angular, Node.js i PostgreSQL. U sklopu razvojnih okvira, korišteni su programski jezici SQL, Javascript, HTML i CSS.

Aplikacija je namijenjena tržištu dugoročnog najma podstanarima te dizajnirana je stavljajući naglasak na iskustvo korisnika pri pretraživanju savršenog prostora te pri stvaranju vlastitog oglasa sa svim potrebnim specifikacijama. Za ovakvu vrstu aplikacije postoji široko i skalabilno tržište ukoliko su ljudski standardi života sve fleksibilniji i podstanarstvo sve više predstavlja najbolje riješene.

U sklopu diplomskog rada, aplikacija je razvijena kao prva testna verzija aplikacije koja se sastoji od glavnih funkcionalnosti oglasnika koje su kroz rad opisane. Evolucija projekta predstavljat će održavanje i poboljšanje postojećih funkcionalnosti, te razvijanje novih funkcionalnosti poput kartnog pregleda prema lokacijama oglasa.

Popis tablica

Popis slika

Literatura

Baze podataka. 2019. https://e-u.hr/dok/udzbenik/31_210.pdf (pokušaj pristupa 20. 7 2019).

Angular. 2019. https://angular.io/ (pokušaj pristupa 14. 7 2019).

Angular material. 2019. https://material.angular.io/ (pokušaj pristupa 15. 7 2019).

Beocaft. 2019. https://beocraft.com/sr/dizajn-web-aplikacija/ (pokušaj pristupa 14. 7 2019).

Figma. 2019. https://www.figma.com/ (pokušaj pristupa 14. 7 2019).

Hostinger. 2019. https://www.hostinger.com/tutorials/what-is-angular (pokušaj pristupa 14. 7 2019).

Inkscape. 2019. https://inkscape.org/hr/ (pokušaj pristupa 14. 7 2019).

Interaction design foundation. 2019. https://www.interaction-design.org/literature/topics/ux-design (pokušaj pristupa 14. 7 2019).

Lucidchart. 2019. https://www.lucidchart.com/pages/uml-use-case-diagram (pokušaj pristupa 14. 7 2019).

Lucidchart. 2019. https://www.lucidchart.com/pages/uml-component-diagram (pokušaj pristupa 15. 7 2019).

Manger, Robert. *Softversko inženjerstvo.* Zagreb: Sveučilište u Zagrebu, PMF Matematički odsjek, 2005.

Material design. 2019. https://material.io/design/introduction/ (pokušaj pristupa 15. 7 2019).

Mountain Goat Software. 2019. https://www.mountaingoatsoftware.com/agile/user-stories (pokušaj pristupa 14. 7 2019).

Nela Dunato. 2019. https://neladunato.com.hr/clanci/brend-logo-vizualni-identitet/ (pokušaj pristupa 14. 7 2019).

Node.js. 2019. https://nodejs.org/en/about/ (pokušaj pristupa 18. 7 2019).

Node.js Dev. 2019. https://nodejs.org/en/about/ (pokušaj pristupa 18. 7 2019).

Pop, Paul Dragos, i Adam Altar. »Designing an MVC model for rapid web application development.« 1172-1179. Elsevier, 2014.

PostgreSQL. 2019. https://www.postgresql.org/about/ (pokušaj pristupa 18. 7 2019).

Pshrmn. 2019. https://blog.pshrmn.com/how-single-page-applications-work/ (pokušaj pristupa 14. 7 2019).

Rodriguez, Alex. »Restful web services: The basics.« U *IBM developerWorks*, 18. 2008.

Tutorialspoint. 2019. https://www.tutorialspoint.com/angular6/ (pokušaj pristupa 15. 7 2019).

Razvoj web aplikacije za iznajmljivanje stanova

Sažetak

Cilj ovog rada je objasniti dizajnerski i inženjerski proces razvijanja web aplikacije za iznajmljivanje stanova. U radu se razmatra motivacija odabira i ciljano tržište, specificiraju se zahtjevi i modeli razvoja te način rada sustava. U sklopu projekta dizajniran je prototip u softveru Figma, složena je relacijska baza podataka u PostgreSQL upravljačkom sustavu, razvijen je prednji ured (eng. front-end) aplikacije za korisnike koristeći Angular okvir, te implementirana je komunikacija između korisničke strane i baze podataka kroz poslužiteljsku biblioteku Node.js u Express okviru.

Projekt je razvijen konceptualno i implementiran. Izvorni kod aplikacije dostupan je na GitHub-u: https://github.com/pbursic/NajamStanova .

Ključne riječi: SPA, web aplikacija, Figma, Angular, Node.js, Express, PostgreSQL, iznajmljivanje stanova

Development of web application for renting apartments

Abstract

The aim of this paper is to explain the design and engineering process of developing a web application for renting apartments. The paper discusses the motivation of the selection and the target market, specifies the requirements and models of development and the system operation. As part of the project, a prototype was designed in Figma software, a relational database was put together in the PostgreSQL management system, a front-end user interface was developed for users using the Angular framework, and the communication between the client side and the database was implemented through the server side library Node.js in the Express framework.

The project was developed conceptually and implemented. The source code for the application is available on GitHub: https://github.com/pbursic/NajamStanova .

Keywords: SPA, web application, Figma, Angular, Node.js, Express, PostgreSQL, renting apartments